AFRL-IF-RS-TR-2004-306
Interim Report
October 2004

AGENTS OVERCOMING RESOURCE INDEPENDENT SCALING THREATS (AORIST)

Altarum

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K275

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** October 2004	**3. REPORT TYPE AND DATES COVERED** Interim Jun 00 – Jul 02

4. TITLE AND SUBTITLE
AGENTS OVERCOMING RESOURCE INDEPENDENT SCALING THREATS (AORIST)

5. FUNDING NUMBERS
C - F30602-00-C-0134
PE - 62301E
PR - ANTS
TA - 00
WU - 01

6. AUTHOR(S)
H. Van Dyke Parunak, Sven Brueckner,
John A. Sauter, and Robert Savit

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Altarum
PO Box 134001
Ann Arbor Michigan 48113-4001

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFTB
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2004-306

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Richard A. Hyle/IFTB/(315) 330-4857/ Richard.Hyle@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

This project uses abstract simulation models of resource allocation and mathematical techniques inspired by statistical physics to study the nonlinear emergent dynamics of distributed decentralized resource allocation. Our techniques seek to characterize the dynamics that may be anticipated in real systems, to predict pathological dynamics such as peaks in required computational effort and catastrophic breakdown in performance, and to develop control methods based on this understanding.

Our general approach begins with a set of abstract Resource Allocation Games (RAG). These games are derived from the Minority Game, a simple model of competition for scarce re-sources that captures essential features of interactions among agents that are heterogeneous, autonomous, boundedly rational, adaptable, parallel, co-situated, and experienced. Our research explores and generally confirms two hypotheses concerning the dynamics of resource allocation.

- The Generality Hypothesis asserts that a generic RAG exhibits dynamics that are intrinsic to resource allocation, in-dependent of mechanism.
- The Specificity Hypothesis asserts that a RAG can be developed to resemble a specific re-source allocation mechanism and study its (idiosyncratic) dynamics.

Final report for this effort published as AFRL-IF-RS-TR-2004-280, October 2004.

14. SUBJECT TERMS
Fine-Grained Agents, Dynamics, Negotiation, Stigmergy, Pheromones, Statistical Physics, Minority Game

15. NUMBER OF PAGES
74

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

1 Summary 1
2 Background 2
2.1 General Approach 2
2.2 Minority Game 3
2.3 Metrics 8
2.3.1 Utility Metrics 8
2.3.2 Load Metrics 9
2.3.3 Input Entropy (Memory Entropy) 9
2.3.4 Output Entropy (System State Entropy) 10
2.4 Overview of Specific Configurations 10
3 Minority Game Core 10
3.1 RAG 11
3.2 Mini-RAG 11
3.2.1 Simplified Structure 11
3.2.2 General Phenomenology 12
3.2.3 Mini-RAG Phase Dependency 14
3.3 Adaptive Mini-RAG 19
3.3.1 N-Learning 19
3.3.2 C-Learning 22
3.3.3 S-Learning 23
3.4 Entropy Analysis 24
4 Logistics Tracks 24
4.1 MICANT-RAG 24
4.2 SNAP 26
4.2.1 Problem Structure 26
4.2.2 Experiments with the ISI Code 26
4.2.3 ISI-RAG 27
4.3 Marbles 29
4.3.1 Problem Structure 29
4.3.2 Difficulty Metrics 30

4.3.3 Marbles RAG ... 33
4.3.4 Is MarbleSize Necessary ... 34
4.3.5 Is MarbleSize Sufficient? ... 35
5 ECM Tracks ... 39
5.1 University of Kansas Track ... 39
5.2 Kestrel ... 40
5.2.1 Algorithm ... 40
5.2.2 Metrics ... 42
5.2.3 Phase Transitions ... 43
5.2.4 Dynamic Attractors ... 48
5.2.5 Parallel with MG ... 49
5.2.6 Control Mechanisms ... 51
6 Technical Leave-Behinds ... 53
6.1 Technical Insights ... 53
6.2 Tools ... 54
7 Transition Efforts ... 56
7.1 Bradley Upgrade ... 56
7.2 Supply Network Engineering ... 57
7.3 Collaboration with ISI/CAMERA ... 58
Acronyms ... 58
References ... 59
Appendix A: Theoretical Analysis of MICANT-RAG (Section 4.1) ... 62
Appendix B: Computation of entropy for the random choice game ... 64

List of Figures

Figure 1: General Architecture of Resource Allocation 2

Figure 2: σ^2/N as a function of $z \equiv 2^m/N$.—System productivity is maximal (and σ^2/N minimal) at a critical value of the size of the agents' strategy space. 6

Figure 3: Domain-Level Metrics 8

Figure 4: Measures of Utility to the Consumer Agent 9

Figure 5: Measures of Load on the Supplier Agent 9

Figure 6: AORIST Project Roadmap.—Ovals are existing simulation code; rectangles are simulation code by AORIST team; rounded rectangles are AORIST code that embodies control techniques, and lozenges are AORIST code that embodies deadline-sensitive techniques. 10

Figure 7: Structure of the general Resource Allocation Game 11

Figure 8: Mini-RAG Results – Mean Award Rate, two suppliers, varying strategy lengths 12

Figure 9: Mini-RAG Results - Std Dev of Group Size, two suppliers, varying strategy lengths. 12

Figure 10: 3-supplier game 13

Figure 11: Entropy-Based Normalization of 3-Supplier Game 13

Figure 12: Standard Deviation of Group Size for Binary (upper curve) and varieties of Partial (lower curves) Rewards 13

Figure 13: Mean Award Rate as Function of Load 14

Figure 14: Standard Deviation of Award Rate as Function of Load 14

Figure 15: Phase Transition in 3-SAT 15

Figure 16: Phase Shift when Load ~ Capacity.—Overall system performance drops dramatically around $N = C$. Each point is the average of 13 runs of 10k turns; $m = 6$. 16

Figure 17: Mean load variance detail.—This plot shows each of the 13 runs at each $N \in \{201,210\}$ for $C = 200$, $m = 6$. At $N = 205$ and $N = 206$, instances of the system are distributed across two distinct states, comparable to the coexistence of two physical phases (e.g., ice and water). 16

Figure 18: Computational effort.—The number of different states over m=6 turns that agents must consider peaks near the transition region (N = 205, 206). Inset: Plot from N<<C to N>>C for C=120. 17

Figure 19: Overlap Region in the Mini-RAG Model 18

Figure 20: Comparison of system state entropy.—Top is actual Mini-RAG result (G = 4, C = 100), center is random baseline per Appendix B, and bottom is Experiment - Baseline. Experimental result is everywhere lower than baseline. 24

Figure 21: Agents in the MICANT-RAG.—Accomplishment of a maintenance task requires coordination among the aircraft being maintained, a maintenance manager, and one or more resources (e.g., parts, workshop space, mechanics), each brokered by a supplier. 25

Figure 22 Oscillations in number of tasks waiting 25

Figure 23: Performance and Effort in CAMERA Baseline.—Both performance and computational load *increase* as constraints *decrease*. 27

Figure 24: ISI-RAG Mean Award Rate (L=15, K=25, N={25,75}, G={25,75}, S=2)................ 28

Figure 25: The Basic Structure of a Marbles Problem (from [5]) 29

Figure 26: Sample Marbles Problems.. 30

Figure 27: Iterative Structure Extension 32

Figure 28: Stupid Solvers on Small (left) and Medium (right) problems.................................... 35

Figure 29: Impact of Pheromone Learning on MarbleSize Algorithm....................................... 37

Figure 30: Impact of Deadline Mechanism on MarbleSize Algorithm 38

Figure 31: Performance and Effort in Case-Based Negotiation.—Number of messages increases, and success de-creases, as *MaxCPUGiveUp* parameter increases.. 39

Figure 32: Phase Shift and Attractors 49

Figure 33: Coloring Patterns in Different Attractors.—Node size is proportional to DOC. 49

Figure 34: Time To Solution as function of Activation Level.. 50

Figure 35: Comparison of ColorRAG and MG 50

Figure 36: Control Mechanisms.—The upper row of figures shows the improvement in *G**DOC (higher is better) under each control mechanism. The lower row shows the change in TTS (lower is better).. 51

Figure 37: Deadline Control in the ColorRAG.—Application of deadline control allows the system to reach a degree of conflict as low as the best results achieved with perfect knowledge (CL = 0)... 52

Figure 38: Result of APSE run.—The large points clustered around the dotted line represent parameter configurations of interest. A complete sweep of the parameter space would have required 121810 simulation runs. APSE found the critical region with only 20046 runs, a savings of more than 6x... 54

Figure 39: AISLE Framework.—Researchers (A) instantiate templates (B) to guide simulation (C), analysis (D), and result presentation (E), generating standardized HTML reports that are integrated with other research documents in an integrated online experimental notebook. Reports include links to the templates (B), ensuring repeatability. The simulations can bedistributed over different processors in a processor farm. 55

Figure 40: Migration Paths among Versions of the Bradley Fighting Vehicle 56

Figure 41: Conflicting Demands in Managing the Bradley.. 56

List of Tables

Table 1: Independent Variables for Resource Allocation Games 3

Table 2: An *m*=3 Strategy 4

Table 3: Information in the Minority Game.—The minority game exhibits four classes of information. Important distinctions concern the *source* (columns) and *location* (rows) of information. 5

Table 4: Mapping ANT Domain Projects onto the RAG 11

Table 5: Extending MG toward the RAG 11

Table 6: Versions of the N-Learning Rule 19

Table 7: *N*-Learning Experiments (*m*=8, *G*=2, *C*=30, NBE=0.9) 20

Table 8: Tilted Consumer Preferences Experiment (*m*=8, *N*=61, *G*=2, *C*=60, Mean over 13 experiments) 22

Table 9: Entities and Parameters for the ISI-RAG 27

Table 10: Parameters for Test Problems 34

Table 11: Enhancements of MarbleSize 36

Table 12: Graph Construction Algorithms. 40

Table 13: Color Change Algorithms 41

Table 14: Correspondences between Mini-RAG and ColorRAG 42

Table 15: Central Configuration 42

Table 16: Parameter Sweeps for Studying Phase Transitions 43

1 Summary

The AORIST project under the DARPA IXO Autonomous Negotiating Teams (ANT) program uses abstract simulation models of resource allocation and mathematical techniques inspired by statistical physics to study the nonlinear emergent dynamics of distributed decentralized resource allocation. Our techniques seek to *characterize* the dynamics that may be anticipated in real systems, to *predict* pathological dynamics such as peaks in required computational effort and catastrophic breakdown in performance, and to develop *control* methods based on this understanding.

Our general approach begins with a set of abstract Resource Allocation Games, or RAG's. These games are derived from the Minority Game (MG), a simple model of competition for scarce resources that captures essential features of interactions among agents that are heterogeneous, autonomous, boundedly rational, adaptable, parallel, co-situated, and experienced. Our research explores and generally confirms two hypotheses concerning the dynamics of resource allocation.

- The *Generality Hypothesis* asserts that a generic RAG exhibits dynamics that are intrinsic to resource allocation, independent of mechanism.
- The *Specificity Hypothesis* asserts that a RAG can be developed to resemble a specific resource allocation mechanism and study its (idiosyncratic) dynamics.

To explore the Generality Hypothesis, we experimented with a generalization of the MG called the "Mini-RAG." This work identified

- persistence of the basic MG phenomenology as specific constraints on that model are relaxed;
- a rich phase structure in the parameter space generated by varying load and capacity;
- the usefulness of system entropy as a metric;
- effort profiles as a function of system size that behave differently than the research community had predicted;
- a variety of control mechanisms inspired by insect pheromones.

To explore the Specificity Hypothesis, we developed versions of the RAG to study systems developed by other ANT program members in both applications domains, Electronic Countermeasures (ECM) and logistics. During the course of the project, we worked with the MICANT team at Vanderbilt, the CAMERA team at ISI, the Case-Based Reflective Negotiation Model (CBRNM) team at Kansas, and the Kestrel team. In both domains our tools and methods were able to

- identify important dynamical behaviors,
- provide improved control; and
- manage system behavior in the presence of an approaching deadline.

AORIST leaves behind a number of important technical insights and tools that have immediate deployment opportunities in critical defense programs. Key technical insights include

- the ubiquity of dynamical phenomena with potential performance implications, including phase shifts and multiple attractors;

- the dependence of computational effort profiles on algorithmic details;
- the power of pheromone learning as a control mechanism;
- the importance of balancing exploration against exploitation as deadlines near.

The AORIST toolkit includes the Adaptive Parameter Search Environment (APSE) for finding regions in parameter space where dynamic irregularities occur, and Automated Instance Sweep for Local Exploration (AISLE) to study those regions in detail.

We are actively pursuing transition of these tools and techniques in three areas.

- We are developing an interactive analysis tool in support of PM Bradley to support decisions and fielding plans for the remanufacture and deployment of various versions of the Army's Bradley Fighting Vehicle. This tool will serve as a platform for successive enhancements based on AORIST technology.
- We are active in two DoD-funded consortia dealing with supply network dynamics, ONR's Supply-chain Practices for Agile Naval Systems (SPANS) and DLA's Defense Sustainment Consortium (DSC). In these projects AORIST techniques will enable defense prime contractors (at present, NorthrupGrumman Newport News and Raytheon, respectively) to improve their performance and reduce costs by managing the dynamics of resource allocation more effectively. AORIST's AISLE tool is already supporting SPANS.
- We are working with ISI to transition some of our enhancements to their algorithms into the deployment channels that they have developed.

2 Background

This section outlines our general approach and architecture, describes the metrics we have developed, and provides an overview of the various configurations we have studied.

2.1 General Approach

The complexities of a real-world system can make it difficult to find and analyze interesting and important dynamics. We begin with an abstract system that is known to exhibit interesting dynamics, and then expand it in the direction of various application systems.

Our starting point is the "minority game," described in detail in the next subsection. Figure 1 and Table 1 show the general structure of this game. A set of consumer agents seek resources from a set of supplier agents. By their choices, they change the load conditions of the various suppliers, thereby impacting the success of their requests for support.

In spite of its simplicity, this model captures the essence of many more complex resource allocation problems, especially when we generalize it to a Resource Allocation

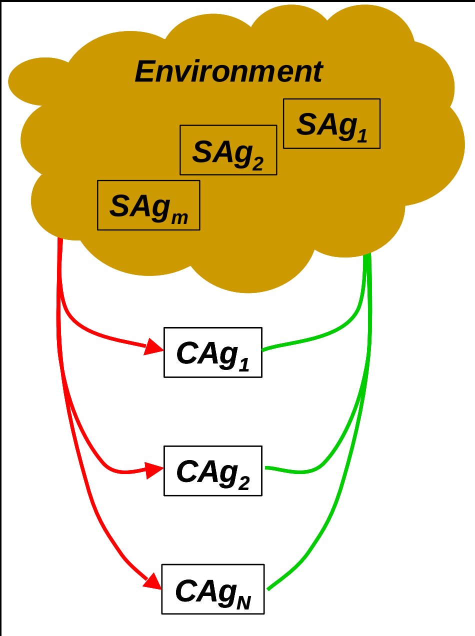


Figure 1: General Architecture of Resource Allocation

Game or RAG (Section 3.2). The agents in this model have a number of characteristics in common with those in more realistic systems. They are

Table 1: Independent Variables for Resource Allocation Games

Parameter	Meaning
G	# of Supplier Agents (SAg's)
C	Total capacity of SAg's, = *i* * *G* for integer *i*
N	# of Consumer Agents (CAg's)
<s>	Vector of strategy parameters used by Cag's (defined in more detail below)

- Autonomous, making decisions locally based on locally available information;
- Heterogeneous, differing in their decision-making criteria;
- Boundedly rational, with strict limits on the reasoning they can perform;
- Adaptable, capable of multiple different behaviors in response to different environmental stimuli;
- Parallel, making decisions without waiting to know the decisions of others;
- Co-situated, sharing a common environment that they influence by their actions;
- Experienced, changing their behavior over time in response to their experience.

Our research explores two hypotheses concerning the dynamics of resource allocation.

- The *Generality Hypothesis* asserts that a generic RAG exhibits dynamics that are intrinsic to resource allocation, independent of mechanism. Testing this hypothesis involves studying (extensions of) the Minority Game and comparing their behavior with that of domain systems.
- The *Specificity Hypothesis* asserts that a RAG can be developed to resemble a specific resource allocation mechanism and study its (idiosyncratic) dynamics. Testing this hypothesis requires developing specific extensions modeled on a particular application.

2.2 Minority Game

AORIST is inspired by a particular dynamic effect first observed in the Minority Game (MG), an abstract model of resource allocation. This model has been extensively discussed in the literature of statistical physics [6, 7, 8, 9, 14], and one Co-PI (Savit) is a recognized leader in the study of its dynamic behavior [17, 18, 19, 27, 28]. The model demonstrates the potential of our technical approach to relate inter-agent and intra-agent behavior and information in support of system characterization, prediction, and control.

The Model.—Consider an odd number, *N*, of agents (representing tasks) and two resources labeled 0 and 1. In terms of the ANT electronic counter-measures domain, an agent might be an ANT representing an incoming target, and the resources might be weapon platforms available to meet those threats. At each time step, each target agent chooses one of the weapon platforms. A platform is overloaded and unable to function if chosen by more than half of the agents. So each target assigned to the less-used platform at a time step receives a point, while each target on the overloaded platform gets nothing. Because agents prefer to be in the minority group, we call this model the "minority game." The best system-wide outcome is a balanced resource load, where the majority and minority populations differ by only one.

In the dynamic decentralized scenarios envisioned by ANT, centralized *a priori* allocation schemes are of little use, and agents allocate resources among themselves in real time, based on information about the current and past utilization of those resources. In the previous section, we described this information as "inter-agent endogenous," "inter-agent" because it is characteristic of the system rather than any single agent, and "endogenous" because it is generated by the agents' interactions rather than imposed externally. This feedback, in which agents make decisions based on a world state affected by their decisions, is a ubiquitous feature of ANT-type resource allocation.

In our simple model, this shared information takes the form of a time series *G* of minority resource identifiers. That is, for any time step *t*, *G(t)* is 1 if resource 1 was less heavily loaded at *t*, and 0 if resource 0 was less heavily loaded. For example, if the minority resource alternates over the first ten rounds of the game, *G* would have the form '0101010101', where new rounds are recorded at the right.

The exogenous intra-agent information consists of a set of strategies assigned to each agent by the programmer. Each strategy tells its agent which resource to choose, depending on the most recent entries in the history *G*. A strategy of memory *m* is a table of 2^m rows and 2 columns. The left column contains all 2^m combinations of *m* 0's and 1's, and each entry in the right column is a 0 or a 1. To use a strategy of memory *m*, an agent observes which were the minority resources during the last *m* time steps of the game and finds that entry in the left column of the strategy. The corresponding entry in the right column contains that strategy's prediction of which resource (0 or 1) will be the minority during the current time step, and thus its recommendation of the agent's action. Table 2 shows an example of an *m*=3 strategy. To apply this strategy to the example ten-step history outlined earlier, the agent extracts the last three digits of the history ('101'), finds that string in the sixth row of the strategy, and chooses action 1. The variable *m* reflects one dimension of the intelligence or sophistication of the agent.

Table 2: An *m*=3 Strategy

m-string	Action
000	0
001	1
010	1
011	0
100	0
101	1
110	0
111	1

At the beginning of the game each agent is randomly assigned *s* (generally greater than one) of the 2^{2^m} possible strategies of memory *m*, with replacement. Thus the dimension of the strategy space occupied by agents with memory *m* is $D = 2^m$. An agent learns a cumulative performance rating for each of its strategies. This rating is an example of intra-agent endogenous information (private to the agent, maintained by the agent's own processing). Following each round of decisions, the agent updates the rating for each of its strategies by comparing that strategy's latest prediction with the current minority group. Then, in choosing its next move, it uses the strategy with the highest performance rating. The agents can choose to play different strategies at different moments of the game in response to changes in their environment; that is, in response to new entries in the time series of minority groups as the game proceeds. For MG, the vector of strategy parameters referenced in Table 1 includes *s* and *m*.

The agents use the time series *G* of past allocations in two different decisions. First, at each time step, each agent updates its estimate of the performance of each of its strategies on the basis of the most recent entry in *G* (the outcome of the previous step). Thus *G* guides it in selecting which strategy to employ. Second, within that strategy, it selects its next move based on the most recent *m* entries in *G*. Interestingly, these functions can be decoupled. Actual performance feedback (inter-agent endogenous information) is necessary in learning the value of different strategies. However, the signal used within a strategy to pick an action can be replaced by a (publicly visible) random variable (thus, inter-agent exogenous information) without greatly changing the system's performance [6]. The primary purpose of this signal is to coordinate actions over the agents, not to convey information about past successes. Distinguishing these two functions is important in mapping real ANT applications onto our model.

This simple model is similar at many points to ANT systems:

- It reflects four categories of information important in understanding ANT dynamics (Table 3).
- Its agents correspond to ANT agents that are seeking resources.
- They are rewarded in a way that reflects the value of constrained resources.
- The agents shape the same environment that in turn guides their decisions.
- They interact through iterated public actions, a form of negotiation.

Characterizing System Dynamics.—The total number of points awarded to the agents reflects the overall productivity of the system, since each point represents a task assigned to a resource that is not overloaded. (In ANT, the more points the system accumulates, the more threats it succeeds in shooting down.) Using methods inspired by statistical physics, we have discovered that this productivity varies with the amount or degree of detail of information used to coordinate the resource choices among the agents (reflected in the dimension of the strategy space). An indirect measure of this productivity leads to a universal form of the dependency.

Consider the time series of the number of agents assigned to resource 1. Because the game is symmetric, the mean of this time series will be close to 50% of the agents. The number of points awarded in a given step is bounded by this mean, since only agents on the minority resource are rewarded. If the variance σ^2 of the time series is small, most minorities will be close to half of the agents, and more points will be awarded over time than if σ^2 is large. So this variance (or equivalently, the variance of the time series of the number of points generated in each time step) will be inversely correlated with system productivity.

The number of agents N turns out to be an important normalization factor. Figure 2 plots σ^2/N as a function of $z \equiv 2^m/N$ on a log-log scale for various N and m. Thus scaled, the data fall on a curve that is universal in N. The horizontal dashed line reflects the value that σ^2/N would take if the agents assigned themselves randomly and with equal probability at each step. Let m_c be the value

Table 3: Information in the Minority Game.—The minority game exhibits four classes of information. Important distinctions concern the *source* (columns) and *location* (rows) of information.

	Exogenous (imposed externally)	Endogenous (generated by the agents)
Intra-agent (within agents)	Strategy space parameter *m* Each agent's strategies	Historical payoff from each strategy Evolved strategies
Inter-agent (between agents)	Can be used to key strategies [6]2]	History *G* of levels of resource utilization

of m at which σ has its minimum. Then, the minimum of this curve is near $2^{m_c}/N \equiv z_C \cong 0.34$, and separates two different phases.

For $z<z_c$ (reflecting low values of m and thus small strategy spaces), the system is in a phase in which system productivity is poor. In this phase agents exhibit a kind of herding behavior. About half the time a large proportion of the agents move to the same resource. Thus on average the agents do worse than they would even by choosing randomly. As m increases to the critical point m_c, overall productivity increases. This part of the curve is not surprising, since one might expect that a larger strategy space (measuring one dimension of increased agent sophistication) ought to lead to improved performance.

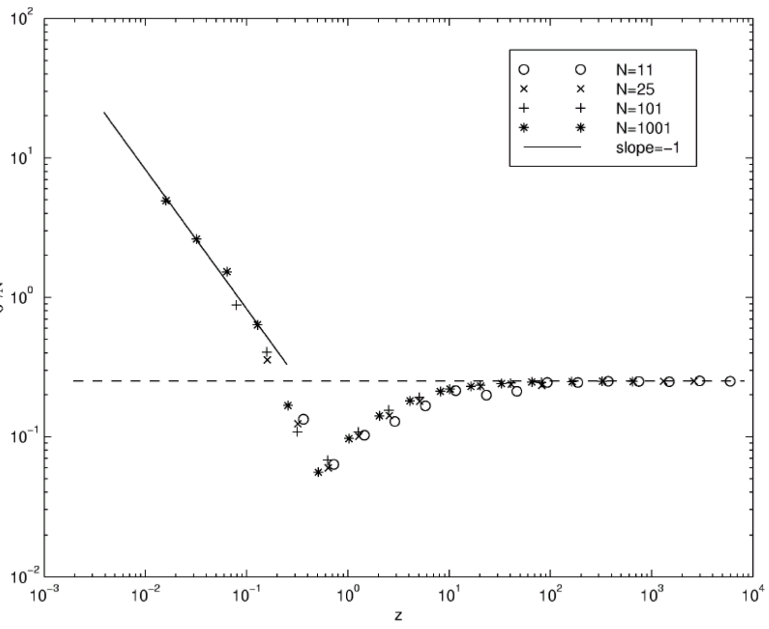


Figure 2: σ^2/N as a function of z≡2^m/N.—System productivity is maximal (and σ^2/N minimal) at a critical value of the size of the agents' strategy space.

We are surprised, though, to find that as z increases beyond z_c, productivity falls off until it approaches that achieved by agents making random choices. That is, the emergent coordination and resulting productivity *decrease* as agent sophistication increases. System performance is greatest when the dimension of the strategy space from which the agents draw their strategies is on the order of the number of agents playing the game.

Because the critical point is dependent on N, the danger of lowered performance away from the critical point is a scaling threat. Traditional scaling threats (due to NP-hard computational complexity) result from the combinatorial explosion of demand on processing and communication resources by a growing agent population. The scaling threat in the minority game is resource-independent, and can appear whether the agent population shrinks or grows from its designed level. Thus our simple model exhibits a *Resource-Independent Scaling Threat* (RIST). In practical terms, if this dynamic is not taken into account, an ANT prototype that performs acceptably on a test problem may underperform when it is fielded on a larger *or smaller* scale.

Around z_c the agents are able to coordinate their choices to achieve a good utilization of resources. By basing their individual decisions on information that they collectively generate, the agents achieve a high level of coordination. This information develops over successive iterations of the system, and it is a form of indirect negotiation among the agents. Thus the performance peak at z_c supports the general thesis of ANT that negotiation can yield coordination. It is important to recognize, though, that this negotiation is happening whatever the values of m and N (and thus z). That is, negotiation alone is not enough to ensure good performance. Even with negotiation, dynamic behaviors such as RIST can compromise system performance if they are not recognized and controlled.

In terms of analogies from statistical physics, the system undergoes a phase transition at z_c. Formally, the existence of such a phase transition in a computational system is comparable to the phase transitions observed in (for example) the actual computational complexity of K-SAT [22]. Operationally, the two transitions have very different implications. Phase transitions that arise combinatorially (such as those in K-SAT) mark regions in which computational costs explode. Well below the transition, most problems can be solved quickly. Well above the transition, most problems can quickly be proven unsolvable. Around the transition, one must expend large amounts of resources before either finding a solution or learning that one does not exist. Thus the phase transition should be avoided in designing and operating a practical system. In contrast, our phase transition marks a "sweet spot" in which system productivity is maximized, and is a target AORIST will enable the system to identify and attain.

Generating and Using Inter-Agent Endogenous Information.—The agents both generate the time series *G* and use it in making their decisions. Our methods (detailed in [19]) permit us to compare the amount of information that they build into *G* with the amount that they can use. This notion of "amount of information" draws directly on the analogy of entropy in statistical physics, and is based on the entropy of substrings of *G* of various lengths. What we discover confirms the importance of the phase transition, and suggests mechanisms for monitoring ANT systems as they operate.

Below m_c, we find that *G* contains no information in substrings of length equal to or shorter than the memory *m* used by the agents. Thus, any agent using strategies with memory (less than or) equal to *m* will find that the portion of the history that it can see contains no predictive information about which group will be the minority at the next time step. The strategies have effectively used all of the information visible to them. Therefore, in this sense, the system is efficient (at least with respect to the strategies). We call this phase *informationally efficient* with respect to the strategies.

Remarkably, *G* does contain information in substrings longer than *m*. This information is generated by the agents' decisions, but is invisible to them. Recognizing the existence of such excess information, we can in principle leverage it for system-level characterization and prediction.

If we repeat the analysis above m_c, we find that *G* contains significant information even in substrings the same length as *m*. In this phase, the strategies are not able to make use of all the information that they can see, and the system is more information inefficient with respect to the strategies. The more *m* exceeds m_c, the more information is left in *G*. Again, we may expect this variation to have considerable impact on the reliability of agent negotiation mechanisms based on microeconomic theory.

The difference between the information efficient and information inefficient phases is seen in the level of success achieved by individual agents. In the high-*m* phase, some agents accumulate significantly more points than they would by simply making random guesses, while others are impoverished. (In the electronic counter-measures domain, some targets recruit more than enough resources, while others are ignored.) However, in the low-*m* phase, no agent ever achieves more than 50% of the possible available points.

We have already noted that performance is not monotonic in the size of the strategy space or agent population, but that moderate populations of moderately sophisticated agents outperform very small or very large populations of very smart or very dumb agents. In addition, the best coordination among agents is obtained when the agents possess only moderate degrees of adaptivity, measured by the number of strategies available to each agent. When each agent has only one strategy, the phase transition disappears, and the game becomes trivial and periodic. As the number of strategies available to each agent increases beyond two, coordination becomes more difficult. The general phase structure still obtains, but the degree of coordination diminishes and the system generally performs more poorly at the transition and in the inefficient phase than it does when each agent has fewer strategies (but still more than one). Thus, in respect both to size of strategy space and to adaptivity, modern research into the minority game exemplifies the exhortation of Horace more than two millennia ago to

hold fast the golden mean,
and live contentedly between
the little and the great.[1]

2.3 Metrics

Experimentation requires well-defined dependent variables. The metrics we have been exploring fall into four categories:

1. Measures of the Utility experienced by the Consumers over the course of the game;
2. Measures of the Load experience by the Suppliers over the course of the game;
3. The entropy of the input information stream available to the Consumers in making their allocation decisions;
4. The entropy of the configurations of Consumers across Suppliers resulting from their allocation decisions.

Figure 3 illustrates an important characteristic of our metrics. For the most part, they concern characteristics that are observable at the level of the problem domain (e.g., load levels, entropy of message traffic). Other approaches to studying the complexity of negotiation-based resource allocation (e.g., SAT-based analyses) are specific to a particular computational implementation, and thus more difficult to generalize.

2.3.1 Utility Metrics

A CAg gets a point at each turn depending on the population and capacity of the SAg it selects. The fundamental time series involved in computing utility metrics is $U_i(t)$, the award to CAg i at time step t. Useful measures on this time series include the various sums, averages, and measures of variance summarized in Figure 4. μ_{U_t} is commonly studied and is called the "mean award rate."

Our experiments typically report the mean, over 13 replications with random seeds, of μ_U and σ_U.

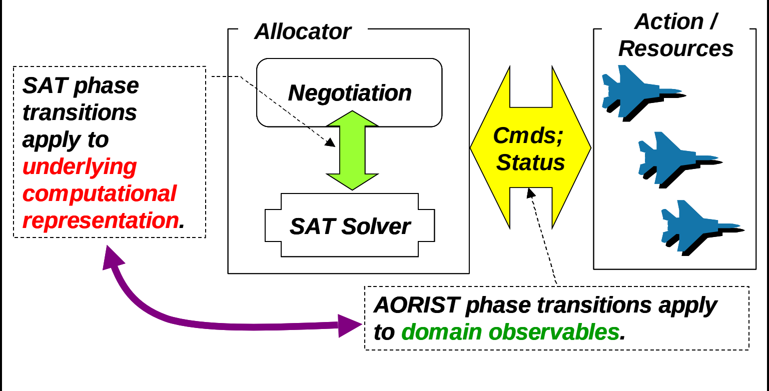

Allocator
Action / Resources
SAT phase transitions apply to underlying computational representation.
Negotiation
SAT Solver
Cmds; Status
AORIST phase transitions apply to domain observables.

Figure 3: Domain-Level Metrics

[1] Horace. *Book ii. Ode x,* Cowper's Translation

Some experiments use a modified version of this metric. At each turn, we record the number of tasks that have *not* received their resource. When tasks both arrive and complete asynchronously, computing the number of outstanding tasks requires summing over the time steps that a task is active. Let $N(t)$ be the number of tasks that are outstanding in time period t. $U(t)$, as defined in Figure 4, already defines the number of tasks that complete in time period t. So the number of outstanding (or waiting) tasks at the end of any time period t is just $W(t) = \sum_{i=0}^{t} N(i) - U(i)$.

$$U_i = \sum_t^T U_i(t); \qquad \mu_{Ui} = U_i / T; \qquad \sigma_U = \langle \sigma(U_i(t)) \rangle_i$$

$$U(t) = \sum_i^N U_i(t); \qquad \mu_{Ut} = U(t)/N;$$

$$U = \sum U_i = \sum U(t); \qquad \mu_U = U/(TN)$$

Figure 4: Measures of Utility to the Consumer Agent

Intuitively, high μ_U maximizes overall system return and increases the probability that an individual CAg is happy. Low σ_U is associated with a more uniform distribution of utility across the population of CAg's.

2.3.2 Load Metrics

The fundamental time series involved in computing load metrics is $L_j(t)$, the number of CAg's that select SAg j at time t. Useful measures on this time series include the various sums, averages, and measures of variance summarized in Figure 5.

Our experiments typically report the mean, over 13 replications with random seeds, of σ_L, $\sigma(\sigma_L)$, and the coefficient of variation ($\sigma_L/(\sqrt{(N/G)})$).Intuitively, high σ_L means suppliers need high capacity for peaks, but must pay for it during valleys, raising overhead costs. It introduces variability that amplifies as one moves down the supply chain (as shown by our previous work on DASCh [25]), and reflects a complex decision landscape for CAg, generally requiring higher decision costs.

High $\sigma(\sigma_L)$ reflects strong sensitivity of system performance on frozen-in strategies.

2.3.3 Input Entropy (Memory Entropy)

For a given m, CAg's make their decisions based on strings of the history of length m. Count all distinct strings of length m in the history, and let p_i be the probability that the ith string appears. The (Shannon) input entropy (memory entropy) is then $S_{in} = - \sum p_i \log(p_i)$. It is sometimes convenient to normalize S_{in} by the logarithm of the number of distinct strings that occur, thus removing the dependency on this number and on the base of logarithms. Intuitively, S_{in} measures the range of uncertainty in the environment in which the CAg's have to act.

$$L_j = \sum_t^T L_j(t); \qquad \mu_{Lj} = L_j / T; \qquad \sigma_L = \langle \sigma(L_j(t)) \rangle_j$$

$$L(t) = \sum_j^G L_j(t); \qquad \mu_{Lt} = L(t)/G;$$

$$L = \sum L_j = \sum L(t); \qquad \mu_L = L/(TG)$$

Figure 5: Measures of Load on the Supplier Agent

2.3.4 Output Entropy (System State Entropy)

For a given G, each step results in a vector of length G specifying the population of each SAg. Count all distinct vectors obtained over a run, and let p be the probability that the ith vector appears. The (Shannon) output entropy is then $S_{out} = - \sum p_i \log(p_i)$. Again, normalization is sometimes helpful. Intuitively, S_{out} measures how unpredictable the aggregate behavior of the CAg's is. Thus we expect it to correspond closely to σ_L. It is relevant to measures of self-organization, and in the endogenous case, to how much the agent's actions contributed to the uncertainty in their environment.

2.4 Overview of Specific Configurations

Figure 6 show our basic experimental structure. Extensions of the MG form the core of our project, and are reviewed in Section 3. These domain-independent configurations permit us to explore dynamics in support of the Generality Hypothesis. Domain- and application-specific extensions of the basic game support the Specificity Hypothesis, and are discussed in Sections 4 (for logistics systems) and 5 (for Electronic Counter-Measures (ECM) systems).

The movement from top to bottom in Figure 6 includes at most four steps.

1. We begin with existing code developed by domain teams (ovals).
2. We develop a version of our RAG to support specific features of the domains system (rectangles).
3. We then add control mechanisms based on the dynamics we observe (rounded rectangles),
4. and finally explore ways to handle the changing pressure of tasks deadlines (lozenges).

The amount of effort invested in each track varied depending on the intrinsic interest of early results and the perceived usefulness of our results to domain teams. Thus some paths do not go through all four steps.

3 Minority Game Core

The MG focuses on dynamics at the critical point where demand and capacity are precisely balanced. We generalized this game to form the RAG, then implemented a subset of the RAG (the Mini-RAG) and explored control mechanisms.

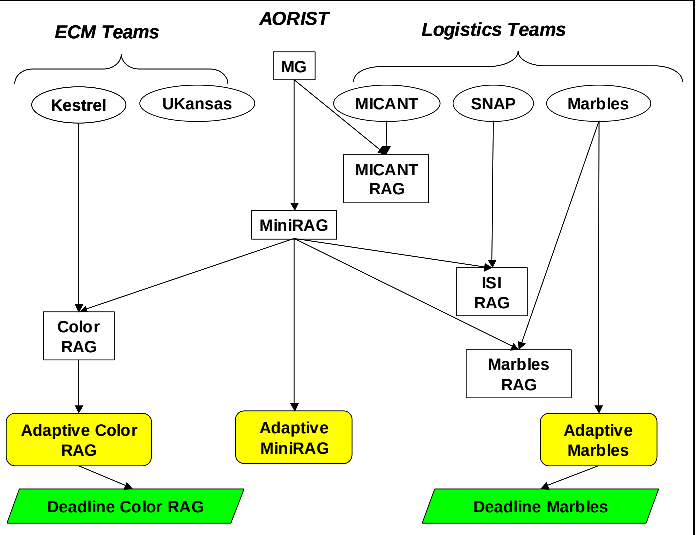


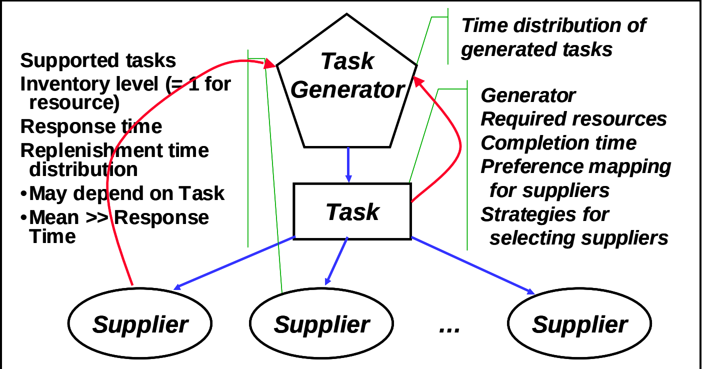
Figure 6: AORIST Project Roadmap.—Ovals are existing simulation code; rectangles are simulation code by AORIST team; rounded rectangles are AORIST code that embodies control techniques, and lozenges are AORIST code that embodies deadline-sensitive techniques.

3.1 RAG

The general Resource Allocation Game (RAG) is a conceptual construct, not an implemented software system. Figure 7 shows its general structure.

- A *task generator* generates tasks over time with a specified temporal distribution.
- Each *task* has a set of required resources, a completion time, a preference over suppliers, and strategies for selecting suppliers.
- Each *supplier* supports some set of tasks. It has an inventory level, a response time and distribution, and a replenishment time and distribution that may depend on the task and in general is much greater than the response time.

Table 4 maps several of the ANT domain projects onto this abstract model.



Time distribution of generated tasks
Supported tasks
Inventory level (≈ 1 for resource)
Response time
Replenishment time distribution
•May depend on Task
•Mean >> Response Time
Task Generator
Generator
Required resources
Completion time
Preference mapping for suppliers
Strategies for selecting suppliers
Task
Supplier
Supplier
...
Supplier

Figure 7: Structure of the general Resource Allocation Game

Table 4: Mapping ANT Domain Projects onto the RAG

	MICANT	CAMERA	Tracker
Task Generators	Aircraft	Pilots Sortie Agent	Targets
Tasks	Repair tasks ≈ MAF's	Missions •Training •Frags •Builds	Targets Tracks
Suppliers	Parts •MALS •Other Squadron •Aircraft W/C •Shops •Tools •Mechanics	Aircraft Pilots Ranges Simulators	Sensors

3.2 Mini-RAG

The full RAG is too complex to be tractable for experimentation. We extended the MG in the direction of the RAG to provide a domain-independent experimental environment.

3.2.1 Simplified Structure

Mini-RAG is an extension of the basic minority game that supports several features of the RAG that go beyond MG. Table 5 details these extensions. The following section highlights our observations under each of these extensions.

Table 5: Extending MG toward the RAG

Feature	MG	Mini-RAG
Load *N* vs. Capacity *C*	N = C+1	*N* and *C* vary independently
Number of suppliers *G*	Only two	Any number of suppliers
Award model	Consumers on an underloaded supplier get a full point; consumers on an overloaded supplier get nothing.	Overloaded suppliers can provide partial satisfaction to agents who choose them.

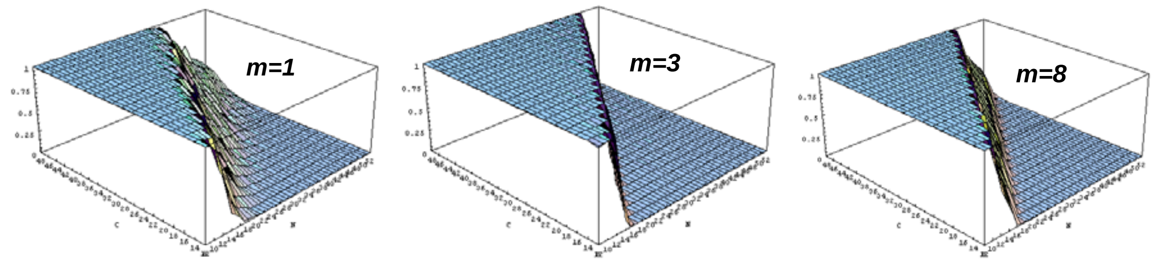


Figure 8: Mini-RAG Results – Mean Award Rate, two suppliers, varying strategy lengths

3.2.2 General Phenomenology

Each of the extensions described in the previous section has interesting effects on the behavior of the system.

Moving Away from N = C + 1.—The MG operates at $N = C + 1$. Figure 8 shows the mean award rate as the capacity (C) and the number of agents (N) vary. The slope near the saturation point (where MG lives) is sharpest for systems that use a history of 3, which is near the minimum for of the Savit curve for this minority game. Both less and more history information stretch out the region where poor results are obtained.

Figure 9 shows the standard deviation of the group size as C and N vary for each of the values of m shown in Figure 8. There are several interesting regions in this space. At the smoother far left and far right corners of the plot there is insufficient information to provide learning, and choices are effectively random. Within the region roughly bounded by $(N+\sqrt{N})/C$ and $(N-\sqrt{N})/2C$ learning occurs. Similar results were obtained when the number of suppliers is increased and when a partial award strategy is employed. A detailed report on the structure of the N vs. C space is available [26].

Exploring G > 2.—Most realistic resource allocation problems involve more than the two resources represented in the (MG). We have studied the dynamics of systems with three and four suppliers. Figure 10 shows the results for three suppliers, scaled using the normalization factors common for MG (as used in Figure 2). The various lines visible in the plot represent different values of m. The plot shows the same general structure as Figure 2 (performance worse than random at the left, asymptotic approach to random performance at the right, and better-than-random performance in between), but the data no longer fall on a curve that is universal in m. One might expect that the problem is due to the base 2 used in the factor $2^m/N$, since we have now moved from two to three suppliers, but a normalization based on $3^m/N$ fares no better.

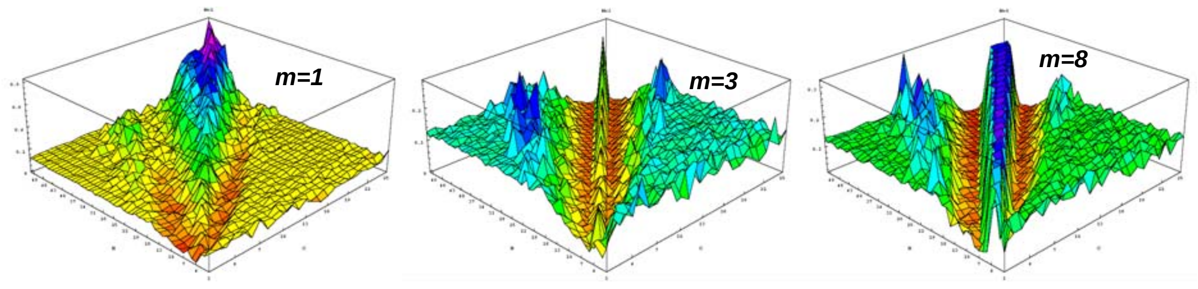


Figure 9: Mini-RAG Results - Std Dev of Group Size, two suppliers, varying strategy lengths

An alternative normalization of the abcissa does restore the universal curve. For a given *m*, agents make their decisions based on strings of length *m*. Count all distinct strings of length *m* in the history, and let p_i be the probability that the *i*th string appears. The (Shannon) input entropy (memory entropy) is then $S_{in} = - \sum p_i \log(p_i)$. Intuitively, S_{in} measures the range of uncertainty in the environment in which the agents act. Figure 11 shows that normalizing the abcissa by e^S/N does restore the smooth curve. This normalization works for the two-supplier case as well. The original $2^m/N$ normalization for that case works because, in the limit in which the probability distribution of *h* is flat, the entropy of substrings of *m*-length is just $\log(2^m)$, so that $e^S = 2^m$.

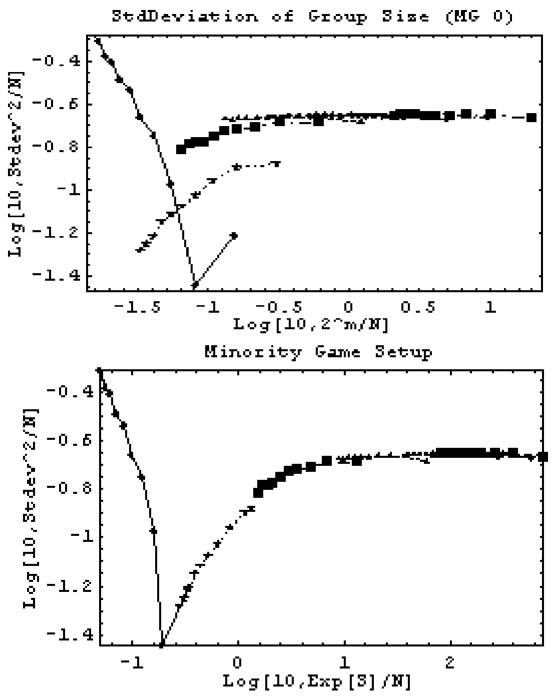


Figure 11: Entropy-Based Normalization of 3-Supplier Game

Variable payoffs.—In each cycle of the standard MG, each agent on the minority resource gains one point, while each agent on the overpopulated resource gets no points. Both of these conditions can be relaxed. Even the minority resource may become less effective as its population increases, leading to models with variable payoffs. In many resource allocation problems, an overloaded resource is not completely shut down, but can satisfy some of the agents that choose it, the partial satisfaction scenario.

In variable payoff games, the amount received by each agent on the minority resource depends on the population of the resource. We have explored two models for this variability. Let *n* be the population of the minority group, and let $r=n/N$. Then, we consider payoff functions of the form $A(r) \sim r^{-\alpha}$, and $A(r) \sim e^{-\gamma r}$, where $A(r)$ is the payoff to each member of a minority group with population $n = rN$. We have also explored payoff functions that depend on the difference in population between the majority and the minority group.

To model partial satisfaction, in addition to giving a point to each agent on the minority resource, we also give an award to a randomly selected subset of the agents on the overloaded resource(s). The size of the subset e is equal to the capacity of the resource. We have experimented with two different values for the amount of the reward. In one case, we give each agent in the rewarded subset a point, just as though that agent had selected a minority resource. In the other, we give each agent in the rewarded subset a fraction of a point, equal to the ratio between the resource's capacity and its population. We have also explored the effect of using fractional awards to rank strategies. Figure 12 shows that the basic structure of the game is invariant over these modifications. The upper curve results from the standard binary reward scheme,

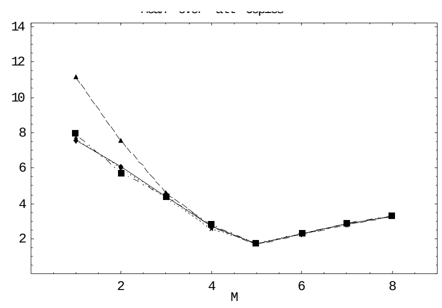


Figure 12: Standard Deviation of Group Size for Binary (upper curve) and varieties of Partial (lower curves) Rewards

while the three possible combinations of partial rewards (partial reward of strategies, consumers, or both) fall together in the lower curve. All curves have the same minimum at the phase transition.

In all these cases, the basic dynamics of the game are unchanged. Performance is suboptimal when *m* is small, approaches the random payoff as *m* grows large, and is better-than-average in the intermediate region.

3.2.3 Mini-RAG Phase Dependency

The general shape of our metrics over the *N* vs. *C* space is strongly suggestive of phase transition signatures in K-SAT problems. To be specific, at m = 3 (near the phase transition for MG), we sweep C from 12 to 48 by steps of 2, and N from 8 to 51 by steps of 1, and explore the landscape of various metrics over this range. Various metrics tend to fall into two categories.

Figure 13 shows the mean award rate (the number of agents receiving a point in each turn) over this space. For N < C, the award rate approaches 1, indicating that the agents are able to distribute themselves without overloading over the available resources. This result is in itself non-trivial, and indicates that the strategy mechanism has learned to avoid overload when the system is operating under capacity. As the load increases and N grows greater than the total system capacity, the award rate falls to zero. Around N = C, the award rate is about 50%, achieving a uniform allocation of resources.

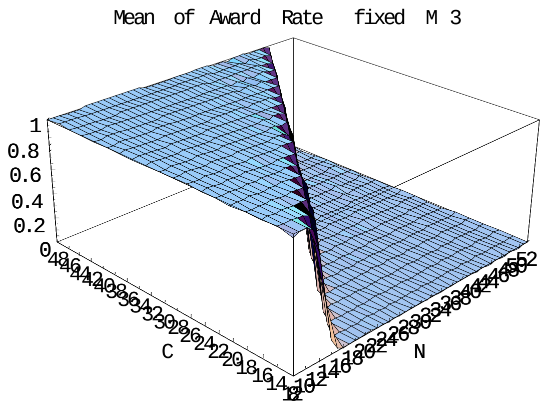


Figure 13: Mean Award Rate as Function of Load

Figure 14 shows the standard deviation of the award rate over this same space. It peaks around N = 25, the configuration of the standard Minority Game. This plot shows that when the system is far from capacity in either direction, its behavior will be relatively stable and predictable. When it approaches the point where capacity and demand are closely balanced, system performance becomes more erratic. Other system metrics have landscapes that mirror one or the other of these two prototypes.

It is instructive to compare Figure 13 and Figure 14 with the lower and upper regions, respectively, of Figure 15 (from [21]). This plot represents a phase transition in constraint satisfaction problems. For a given number of variables *N* and number of clauses *L*, an instance of random 3-SAT is produced by randomly generating *L* clauses of length 3. Each clause is produced by randomly choosing a set of 3 variables from the set of *N* available, and negating each with probability 0.5.

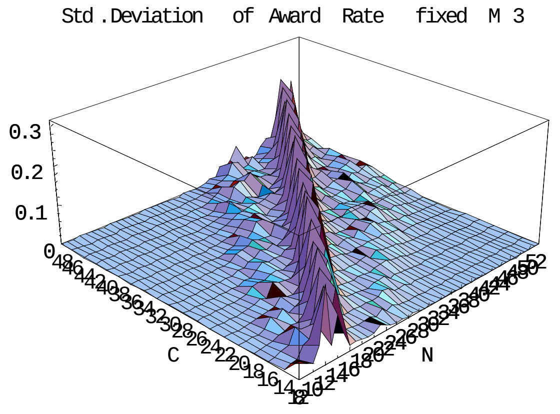


Figure 14: Standard Deviation of Award Rate as Function of Load

Two metrics in this system are of interest: the number of calls to the atomic constraint resolution procedure needed to determine whether a given problem instance is or is not satisfiable (reflecting the computational complexity of the given problem instance), and the probability that such an instance is indeed satisfiable (estimated as the fraction of a population of randomly generated instances that are found to be satisfiable). Figure 15 plots these two metrics as a function of *L/N*.

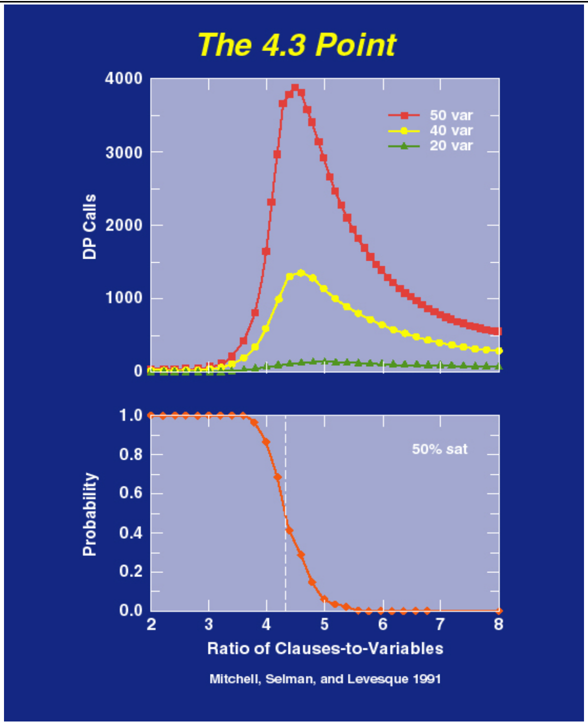


Figure 15: Phase Transition in 3-SAT

The lower plot shows that as the number of clauses increases for given *N*, the probability that the problem is satisfaction drops. This behavior is expected: as additional clauses are added, the likelihood increases that the problem will be over constrained.

The upper plot shows that the computational effort needed to determine whether an instance is or is not able to be satisfied is reported as the median of 500 trials is low for instances with either very few or very many clauses, and peaks around *L/N* = 4.3.

The computational effort peaks at the point where the probability of satisfaction is 50%. If almost all instances are either able to be satisfied or not, it doesn't take long to find either a solution or an irreconcilable pair of clauses. However, in the intermediate region, a much larger proportion of the domain must be examined.

The similarity between the two systems is important. Both cases distinguish a region in which solutions are abundant and easy from a region in which they are rare. In both cases, the transition between these two regions is marked by a peak in an undesirable system measure (variability in award rate in the Mini-RAG; calls to the solver in 3-SAT). The similarity suggests that this signature is not an artifact of the indirect negotiation used in the MG, but reflects a general property of any system that is close to its capacity. In particular, we expect that whatever negotiation mechanism is used to address a resource allocation problem, its behavior will degrade in some observable way in the region where load is about equal to capacity. The way in which this degradation is manifested may depend on the mechanisms being used. In some systems, it may take the form of high variability in the results achieved. Other systems may achieve low variability, but at the expense of more protracted negotiations. The point is, a closely constrained environment has intrinsic dynamics, and the flow of information between the environment and the agents that interact with it will impose its dynamics on those agents, whatever the negotiation mechanisms being used.

Importantly, pressures in real-world systems will tend to drive the system toward the region of high cost. The cost of resources tends to reduce *C*, while demand for system execution tends to increase *N*. Thus over time, many systems will find themselves along the dynamically complex region near the MG.

The plots presented thus far are the mean of 13 experiments at each assignment of experimental parameters, each experiment only varying by the initial random number seed. By plotting the results of each experimental run separately it becomes clear that the steep side slopes of the transition region actually represented to distinct phases of system behavior.

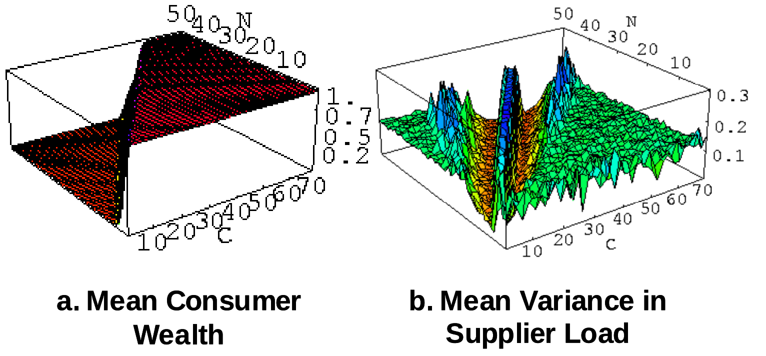


Figure 16: Phase Shift when Load ~ Capacity.—Overall system performance drops dramatically around $N = C$. Each point is the average of 13 runs of 10k turns; $m = 6$.

Figure 16 represents the performance of the Mini-RAG as a function of the number of agents N requesting resources from suppliers that have a total capacity of C, and the variance in the supplier load. A lower variance results in better system performance (since the load is more evenly balanced and more agents are satisfied).

In this discussion, we focus on the region near $N = C$, where σ^2/N peaks. Figure 17 shows a slice at constant $C = 200$ through this peak. (We show only the flank on the side where $N > C$; a theorem proven in [26] shows that this curve is symmetrical about $N = C + 1$). This figure plots the result of each run of the system separately, in contrast with Figure 16, which plots the average at each value of N and C. In the region corresponding to the flanks of the central peak in Figure 16b, the values of σ^2/N for different runs separate into two quite distinct groups. For a given C and a range of N, there is a well-defined coexistence region between two very distinct phases in these resource allocation games, comparable to the coexistence of different phases in a physical system. The values of σ^2/N in Figure 16b at the top of the central peak and at the bottom of the neighboring valleys accurately reflect typical behavior of individual runs. But the average values along the flanks are atypical; any given individual run belongs either to the high or the low phase. This system thus exhibits not one but two phase changes, one on each flank of the system.

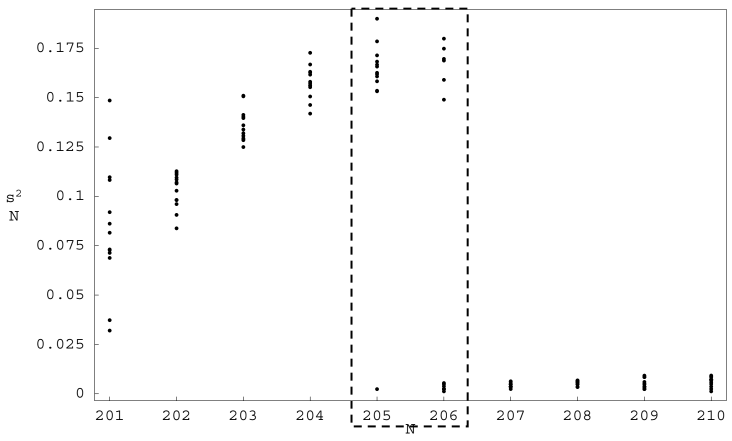


Figure 17: Mean load variance detail.—This plot shows each of the 13 runs at each $N \in \{201,210\}$ for $C = 200$, $m = 6$. At $N = 205$ and $N = 206$, instances of the system are distributed across two distinct states, comparable to the coexistence of two physical phases (e.g., ice and water).

What is the distribution of computational effort as one moves through this phase change? Figure 18 plots the total number of rows of the strategy table that were consulted for each value of N along the same $C = 200$ line used in Figure 17. This represents the total number of unique supplier states (overloaded or underloaded) over m=6 turns that the agents have seen. These histories are used by the agent's strategies to select the supplier in the next turn. This value peaks just over the coexistence region identified in Figure 17, and drops away elsewhere. The inset (for a system with $N = 61$) shows that it is as low for $N >> C$ as it is for $N << C$.

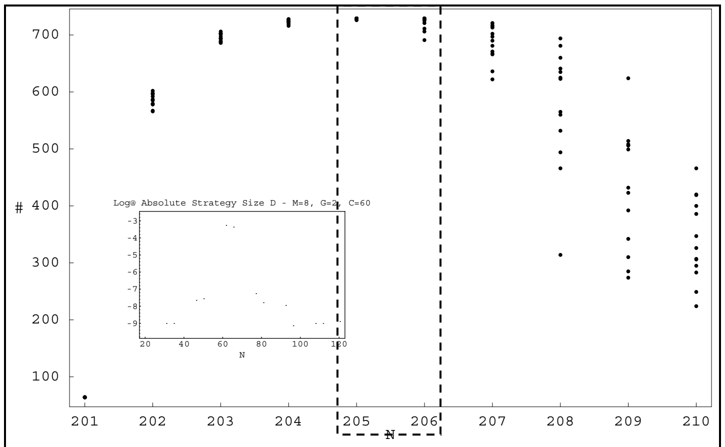


Figure 18: Computational effort.—The number of different states over m=6 turns that agents must consider peaks near the transition region (N = 205, 206). Inset: Plot from N<<C to N>>C for C=120.

It has been a commonplace in the constraint analysis community that constraint satisfaction problems exhibit Easy-Hard-Easy effort profiles, while constraint optimization problems exhibit Easy-Hard profiles. The Mini-RAG is an optimization problem, but exhibits an overall Easy-Hard-Easy profile. We explore this discovery further in [24].

We can develop an intuition supporting this pattern of computational load. The peak near the minority game, and the valleys on either side, represent qualitatively different dynamics in the game, resulting from differing numbers of states accessible to the system. At $N = C+1$ (201 in Figure 18), due to the structure of the game, only 64 different states over m=6 turns are possible. This results in the transient dip in the load profile at N=C+1. Near N » C+1, there are a total of 729 different states over m=6 turns. As one moves into the valleys, the probability increases that the system will see fewer of the possible states since either both suppliers will always be overloaded (for $N >> C$) or underloaded ($N << C$) over most or all of the m=6 turns. The ability of a given population of agents to distinguish these states depends on the distribution of strategies (generated in our case randomly and fixed for the duration of the game).

The load represents the entire history of the experimental run. The σ^2/N plots in Figure 16b just show the variance near the end of the run after the population has settled into a pattern. Both the upper and lower populations experienced nearly all the possible states during the game. At some point in the game, for six consecutive turns, both suppliers were overloaded. Strategies of agents in the lower band were able to respond to this history with choices that maintained the supplier loading balance (keeping both suppliers overloaded). Strategies of the agents in the upper band were not able to maintain this balance and hence that history did not drive the system to a stable regime, rather the agents continued to generate historical states that included both underloaded and overloaded suppliers resulting in a higher loading variance.

In Mini-RAG, the critical feature driving the phase transition is the diversity among the agents and their strategies. Some of this diversity is frozen into the agents through the randomly generated strategies, and some of it emerges as the agents dynamically learn to prefer one strategy to another and so distribute themselves over the problem space. The emergent dynamics behave differently on the peak and in the valley, and in between they take longer to converge.

Near the point where the demand on the system balances the system's global capacity, we observe a transition in the system's dynamics, in one of our chosen metrics, the probability of a consumer to gain a resource. Thus, for the purpose of controlling the performance of the system on the basis of its known dynamics, it is important to understand the conditions, under which a particular system settles on either phase in the overlap region. We initiated a set of experiments and analyses to explore this essential question.

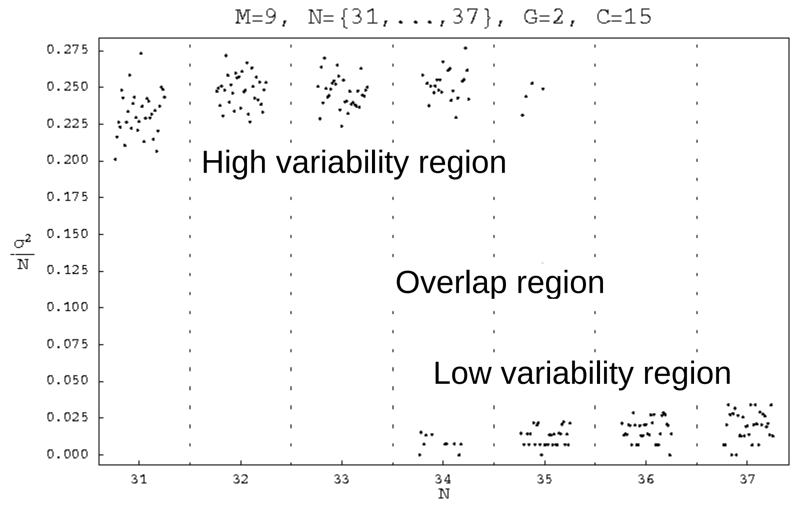


Figure 19: Overlap Region in the Mini-RAG Model

At certain configurations of the Mini-RAG's main parameters *M* (length of accessible history), *N* (number of consumers), *G* (number of suppliers), and *C* (capacity of any individual supplier), the non-deterministic nature of the agents' decision processes drive the system into one of two fundamentally different phases. Figure 19 shows an example of the phase change, plotting the "Standard Deviation of the Group Size" metric (σ^2/N) for 32 individual experiments with M=9, N={31,…,37}, G=2, and C=15 (note that demand is always greater than capacity). To enhance the readability of the diagram, we plot the individual experimental result for a particular value of N offset around the actual integer value on the horizontal axis.

In this particular experiment, systems with N=34 or 35 consumers may end up either in the variability region (high σ^2/N), or in the low variability region (low σ^2/N) of the system dynamics. In the Mini-RAG, a high variability means that on average there is a poor allocation of resources since in any run of the game there is a high probability that the number of agents is not evenly balanced on the two resources. We investigated the hypothesis that the strategy tables of the consumer agents determine which region a particular instance of the game will occupy. Each consumer has two randomly chosen strategy tables that define for any possible input string of M*G bits (the state of each supplier in the last M cycles), the supplier that the consumer should select for the next round of the game. The possible input states and their responses are rows in a strategy table. Each strategy table is scored based on how well the agent would have performed had it chosen that strategy table. In each cycle a consumer will follow the suggestion of the strategy table with the highest score and break ties among strategies randomly. Thus, the entries in the various strategy tables determine the behavior of the consumer population.

To verify our hypothesis, we used strategy tables from systems that ended up in the high or low variability region in the experiment shown in Figure 19. These were then used as the starting strategy tables for another run of experiments. Under several different variations, the configuration of the strategy table was only a minor determinant in the final state of the system.

The key finding of this experiment is that it is not possible to choose a single strategy at the outset of system that is able to guarantee that the system will end up behaving in a desired fashion when it nears its capacity limits. Thus static approaches to solving the problem (such as finding the "best" design of a system) will fail to address the complex dynamics that they are subject to. We focused the rest of our efforts on dynamic methods that can adapt to the system based on its performance.

3.3 Adaptive Mini-RAG

Next we chose a learning approach to adapt to the state of the system. The purpose of our local adaptive rules is to influence the dynamics of the Mini-RAG model at runtime, based on the currently observed operation of the system. To cope with the inherent uncertainty and incompleteness of information and the dynamic nature of the resource allocation process we instantiate a generic forgetting process for each probabilistic Mini-RAG control rule. The forgetting process is inspired by the truth maintenance mechanism in marker-based stigmergic systems, such as social insect colonies [4, 23]. In these systems, the current local concentration of highly volatile chemical substances (pheromones) previously deposited by agents (e.g., insects) influences the outcome of probabilistic decisions of other agents, who, in turn, may deposit pheromones themselves. The continuous evaporation (proportional reduction of local concentration) of the pheromones ensures that any information that is not constantly maintained (new deposits) automatically vanishes from the system's memory.

In the following sections we describe a number of adaptive mechanisms that we designed to influence the emerging dynamics of our Mini-RAG model at runtime. These mechanisms are based on probabilistic decision rules, whose probability values are decreased by the automatic truth maintenance mechanism and increased by learning rules. We explore dynamic adjustments to the number of consumers *N*, the number of suppliers *C*, and the use of strategies *S*.

3.3.1 N-Learning

In the Mini-RAG model, each consumer must bid for a resource in each cycle. In a situation where there are many more consumers than resources in the system, this requirement may drastically reduce the probability of success for the individual consumers (Mean Award Rate metric), depending on the chosen award policy (binary or partial satisfaction). In other resource allocation systems, having a large number of agents bidding for limited resources significantly increases the amount of time the system requires to negotiate a solution. A standard approach to address this problem is to drop some agents out of the bidding process early on, so the other, more important demands can be met without swamping the system in negotiations that are doomed to fail.

We introduce an adaptive decision rule that allows the individual consumer to decide in each cycle, whether to actually send in a bid or not. The No-Bid rule is based on a No-Bid Probability (NBP) value in each consumer agent, which is decreased by the automated truth maintenance process using a system-wide No-Bid Evaporation (NBE) factor.

Table 6: Versions of the N-Learning Rule

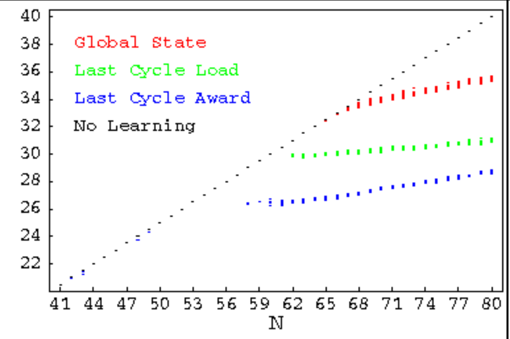
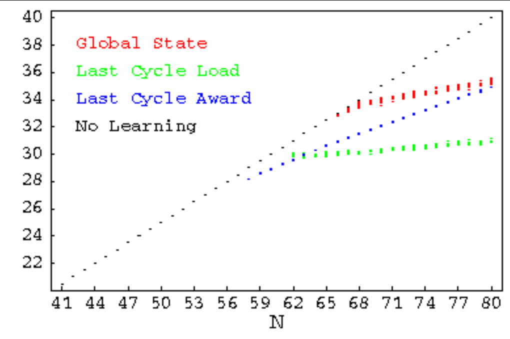
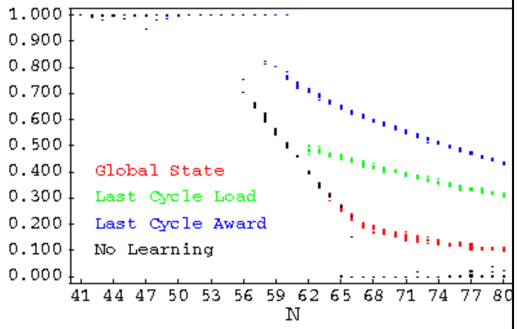
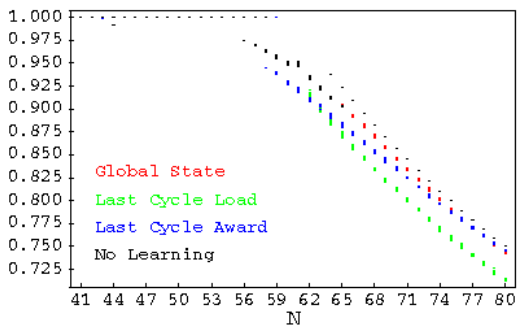
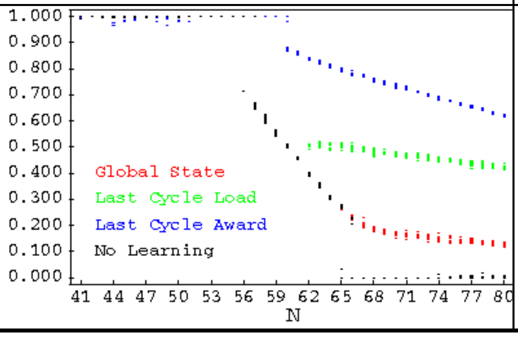
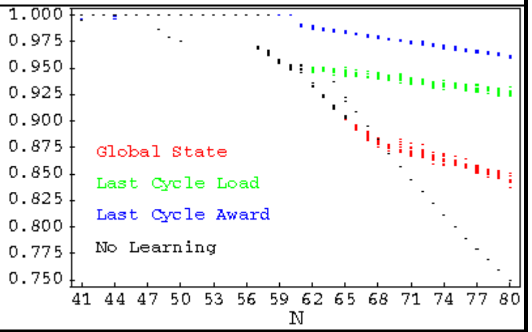
Name	Description
Global State	A consumer increases its No Bid Probability (NBP) value if all G suppliers had been overloaded during the last M cycles. This rule requires global information on all suppliers.
Last Cycle Load	A consumer increases its NBP value if all G suppliers had been overloaded in the last cycle. This rule requires global information on all suppliers.
Last Cycle Award	A consumer increases its NBP value if the bid it had submitted during the last cycle did not return a full unit of the requested resource. In difference to the other rules, this rule increases the NBP value by the missing portion of the resource (1 - consumer award). This rule requires only local information.

The learning rule, which increases NBP by 1-NBE, fires when it perceives a situation where too many consumers bid for resources. We experimented with three versions of this rule, which use consecutively less global information. Table 6 specifies these variants.

The plots in Table 7 show the results of a representative experiment that we performed with the various N-Learning rules. We configured our system to contain two suppliers (G=2) with thirty resources (C=30) each and we varied the number of consumers (N) from 41 to 80. In all experiments we allowed the consumers to use the load history of the last eight cycles (M=8) in their strategy decisions. We executed 13 identical copies with different random seeds, using binary and partial satisfaction policies and applying all three variants of the N-Learning rule as well as a no-learning experiment (uncontrolled Mini-RAG) for comparison.

We plot the Mean Group Size and the Mean Award Rate metrics for the different learning rules and award policies. As we can see in the Mean Group Size metric, N-Learning in general reduces the number of bidding consumers. Comparing the three learning rules, we also notice that the ef-

Table 7: *N*-Learning Experiments (*m*=8, *G*=2, *C*=30, NBE=0.9)

Metric	Binary Satisfaction	Partial Satisfaction
Mean Group Size		
Mean Award Rate		
Effective Mean Award Rate		

fect of the Last Cycle Award rule begins at smaller N than the more global rules. This difference can be explained by the fact that none of the global rules can fire before there are enough consumers to overload both suppliers (N>G*C), while only one overloaded supplier is required to fire the local rule.

In the case of the binary award policy, the Last Cycle Award rule generally leads to a larger number of consumers dropping out (highest average NBP) followed by the Last Cycle Load rule, and finally the Global State rule. This order is not surprising, since it is more likely that only one supplier is overloaded than both suppliers being overloaded once; and having both suppliers being overloaded eight (M) times in a row is even less likely. But, the more likely the firing of an N-Learning rule is, the higher (on average) will be the No-Bid Probability (NBP). For extremely overloaded systems (not shown here), the probabilities of all three events approach 100 percent.

The Mean Award Rate metric tells us the probability of a consumer to successfully acquire a resource in an average cycle. All N-Learning rules are as good or better than the uncontrolled Mini-RAG when the capacity is greater than demand. We also find that the Last Cycle Award Rule outperforms the Last Cycle Load rule, which is still significantly better than the Global State rule. The Last Cycle Award Rule performs the closest to the optimal behavior, which is C/N for N>G*C. This rule is the most aggressive in its learning, and thus tends to drop out more consumers.

In the case of partial satisfaction, we see the same Mean Award Rates in the first phase. But the second and third phases differ significantly. In the third phase, both suppliers are constantly overloaded, which guarantees, that all 60 units of resource are shared among the consumers. Thus, for the no-learning case we see a decrease of the Mean Award Rate that follows G*C/N – the maximum possible award. In the second phase one of the two suppliers sometimes receives fewer bids than it has capacity and thus some resources go unused. Therefore, we see a less than optimal average system performance.

For the Mean Award Rate metric all the N-learning rules perform worse than no-learning. This is to be expected. To get the highest award rate, you need to have all the consumers bidding since then they will all receive at least a partial award. If you drop out any consumers, then you decrease the total award possible and hence the Mean Award rate metric (which is the mean among all consumers even those not bidding). The N-learning rules perform better than no-learning on a different Mean Award Rate metric we call the Effective Mean Award Rate. The Effective Mean Award Rate is the average award taken over just the consumers participating in the bidding. As can be seen in Table 7, all three N-learning strategies perform better than no-learning on this metric with Last Cycle Award outperforming Last Cycle Load which outperforms Global State. This is the desired behavior. Everyone does equally poorly when you include all the consumers. Dropping out some lower priority consumers means that higher priority consumers can receive a higher average award than otherwise possible.

We draw two major conclusions from our discussion of the *N*-Learning approach to the control of the Mini-RAG:

1. Reducing the effective load (i.e. the number of consumers bidding) on a binary award resource allocation system has the potential of improving the system's performance. The more aggressive learning strategies perform much better than the less aggressive strategies.

2. Reducing the effective load on a partial award resource allocation game can improve the performance of the consumers that do bid at the expense of reducing the total award for the system.

In all cases, it is important to understand the metrics that make the most sense for the application at hand. Once that metric is known, appropriate N-learning strategies to optimize for that metric can be developed.

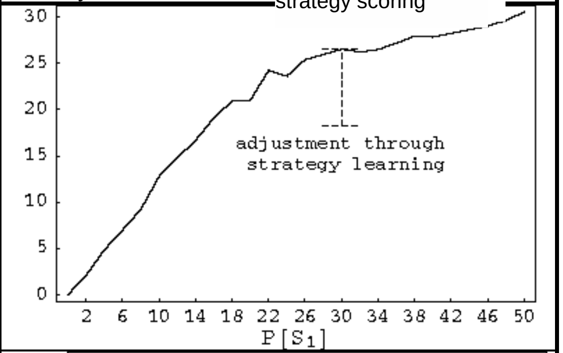
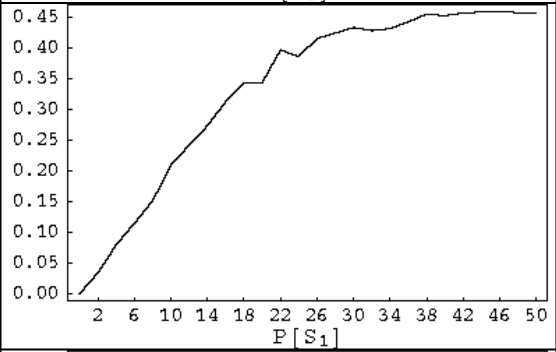
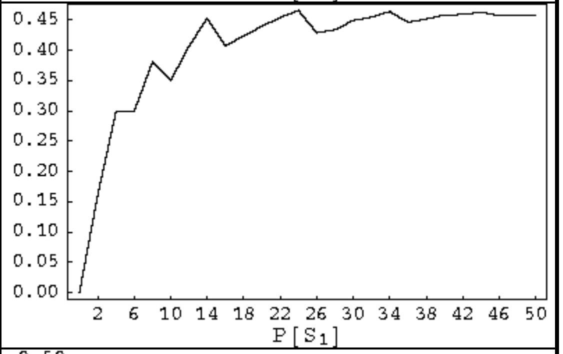
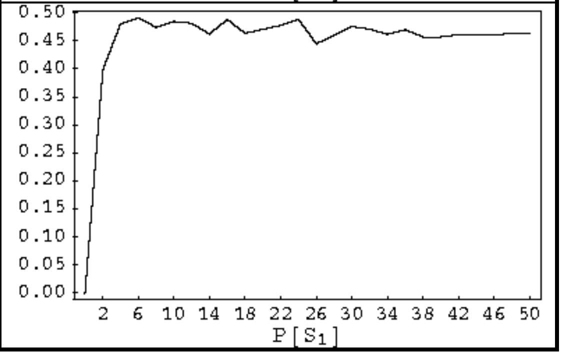
3.3.2 C-Learning

The consumers in the Mini-RAG model tend to have equal preference for each supplier. This uniform distribution of consumer preferences cannot be guaranteed in more realistic resource allocation scenarios. Rather, driven by various influences (e.g., past experience, geographic location, pricing strategies), there will be a distinct preference mapping of consumers to suppliers. In an attempt to reproduce this aspect of real resource allocation systems, we extend our Mini-RAG model, assigning each of the *G* suppliers a probability, with which a consumer's strategy will select the supplier. As a consequence, we observe an average load distribution over the population of suppliers that reflects these probabilities.

Non-uniform consumer preferences create a new potential for sub-optimal system performance and thus require an additional adaptive process. The optimal utilization of resources no longer depends on the balance between system wide capacity and consumer load alone. Rather, the distribution of the capacities across the supplier population must match the non-uniform consumer preferences.

The second plot in Table 8 illustrates the decreasing performance (Mean Award Rate) as the consumer preferences become more and more tilted

Table 8: Tilted Consumer Preferences Experiment (*m*=8, *N*=61, *G*=2, *C*=60, Mean over 13 experiments)

Metric	Binary Satisfaction Adjustment through strategy scoring
Mean Group Size S_1	
Mean Award Rate	
Mean Award Rate Manually Adjusted Capacities	
Mean Award Rate C-Learning	

(as P[S_1], the probability of selecting Supplier 1, approaches 0). We also observe that the strategy scoring mechanism of the uncontrolled Mini-RAG still is capable of offsetting the preferences in the supplier selection to create less imbalance than the consumer preference settings would have generated. This is shown as the difference between the dotted line and the observed Mean Group Size of Supplier 1 in the top plot.

The plot in the third row of Table 8 show the results of an experiment in which we tilt the consumer preferences, but this time we manually adjust the supplier capacities to match the imbalance in the consumer preferences. As expected we find an increased performance on the Mean Award Rate metric when P[S1] < 25% compared to the non-tilted capacities. Still there is a drop off in performance for P[S1]<10%. We suspect that finite size effects in the probabilistic choice of suppliers by the strategies are the cause of the drop-off in performance.

We designed an adaptive mechanism (*C*-Learning) that enables the individual suppliers to change their capacity by trading resources with the goal to match the perceived consumer preference distribution. In the *C*-Learning mechanism, suppliers trade resources one unit at a time based on whether they were overloaded or underloaded in the previous cycle. *C*-Learning does not change the total resource capacity of the system, it just seeks to find an optimal balance of these resources across the supplier population.

The plot in row 4 of Table 8 shows the results with *C*-learning. The adaptive results are slightly better than the fixed ones, even though the fixed experiment distributed the resources to exactly match the consumer preferences. We explain this improvement by the fact that the consumer preferences are probabilistic, and during the run of a particular experiment, the random choices of the strategies may differ from their expected mean. With the adaptive rules the suppliers remain capable of adjusting their resources exactly to the emerging consumer preferences and thus improve the performance. This advantage of the adaptive approach is especially visible in the performance of the extremely unbalanced systems, where the limited size effects distort the preset consumers' preference distribution.

The two main conclusions we can draw from the *C*-Learning are:

1. Local resource trading among individual suppliers can optimize the global distribution of resources to cope with non-uniform consumer preferences.
2. Adaptive mechanisms have the potential to outperform static configurations set to the optimal average value.

3.3.3 S-Learning

S-Learning is another promising adaptive learning mechanism. While *N*-Learning and *C*-Learning modify the effective load and the effective capacity of the resource allocation system, the external configuration parameters of the system, *S*-Learning focuses on the internal decision processes of a consumer.

In the uncontrolled Mini-RAG model, each consumer has *S* strategies (in all our experiments *S*=2). A new row is added to each of the strategy tables each time a new historical state (bit string with the load history of the *G* suppliers over the last *m* cycles) is encountered that was not already present in the strategy tables. Any future occurrence of that historical state produces the same response from the strategy table, regardless of the global dynamics of the system.

Additional adaptivity in the strategy selection will permit the system to cope with changing dynamics. Such changes occur, for instance, when the system moves from the "scarce resources" phase to the "limited resources" phase, or when the distribution of resources in the system has changed. A set of consumer strategies, which had gained dominance (larger wealth) over other strategies in one phase, may not provide a suitable response in the new phase. To adapt to these changes, the wealth discrepancies among the strategies of a consumer could be continuously eroded over time. The addition of this truth maintenance mechanism allows the consumer to adapt its selection of strategies as the environment changes. It is also possible for a consumer to change its response to a state it encounters. Adaptivity at the level of rows in the lookup table greatly increases the granularity of the strategy learning mechanism of a consumer, who in the uncontrolled Mini-RAG model had to rely solely on the "intelligent" choice between the *S* strategies.

3.4 Entropy Analysis

A persistent question throughout studies of the Mini-RAG is the degree to which the results observed differ from random behavior. To study this question in a disciplined way, we analyzed the entropy over states of the system (output entropy, Section 2.3.4) under the hypothesis of random actions (Appendix B), and compared the result with experiments. The results (exemplified in Figure 20) show a clear deviation from the random baseline in both the magnitude and structure of the results.

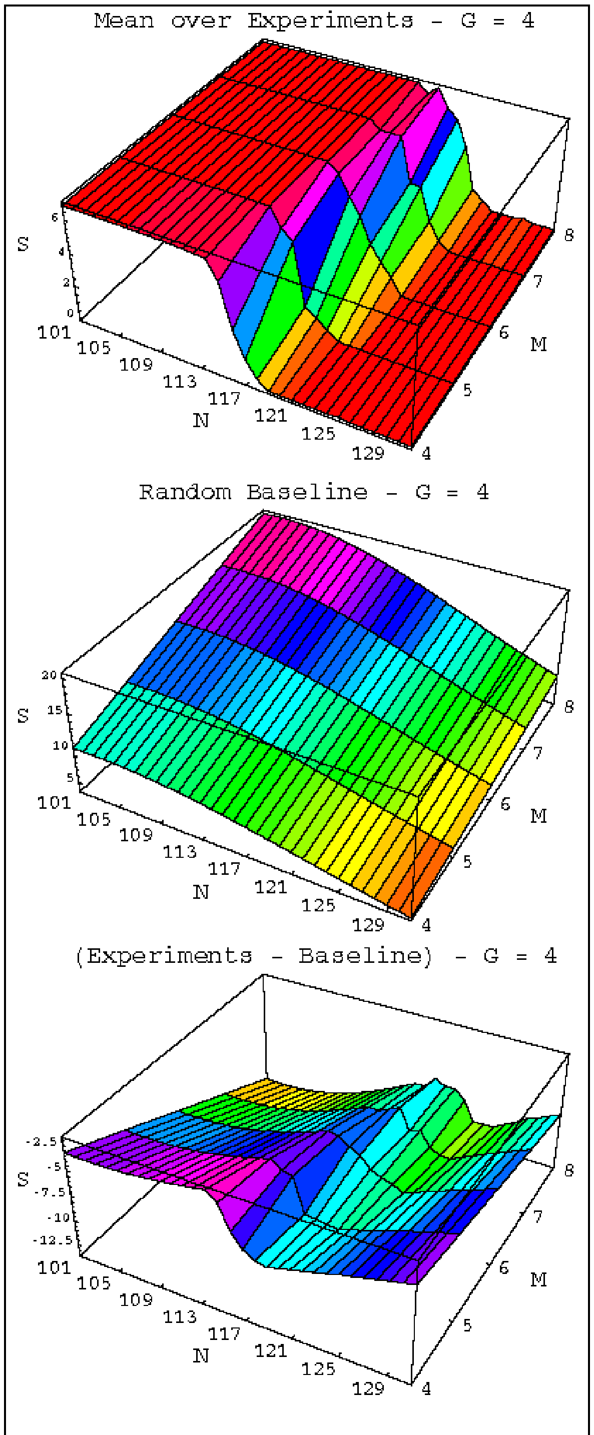


Figure 20: Comparison of system state entropy.—Top is actual Mini-RAG result ($G = 4$, $C = 100$), center is random baseline per Appendix B, and bottom is Experiment - Baseline. Experimental result is everywhere lower than baseline.

4 Logistics Tracks

4.1 MICANT-RAG

The MICANT team led by Vanderbilt University is developing a system for scheduling aircraft maintenance. We studied their system and implemented the architecture shown in Figure 22. In this simple game, a variable number of aircraft agents generate maintenance tasks with variable frequency. Tasks request resources through a single Maintenance manager. The Maintenance Manager maintains a preference mapping for the suppliers that supply parts, labor, and bay space for the task. The Maintenance manager randomly requests resources (weighted by the preference mapping) among the possible suppliers. If it fails (because the supplier is already busy with a task), the maintenance manager re-tries after a timeout. With task loads that are non-saturating, the number of waiting tasks increases asymptotically with an oscillation of constant period that persists indefinitely (see Figure 21).

Our experiments with this system vary the task generation rate and resource recovery rate, and look at the number of tasks awaiting access to resources.

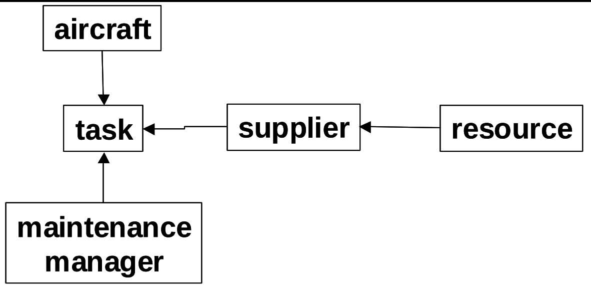


Figure 22: Agents in the MICANT-RAG.—Accomplishment of a maintenance task requires coordination among the aircraft being maintained, a maintenance manager, and one or more resources (e.g., parts, workshop space, mechanics), each brokered by a supplier.

In the first round, tasks arrive at a fairly constant rate, and need three kinds of resources to be executed (labor, space, and part). The plots for the number of tasks concurrently negotiating for resources show several interesting features when the task generation rate is close to the recovery rate:

1. There is a fine-structure oscillation with a period on the order of 160k cycles.
2. The number reaches an asymptote.
3. The approach to the asymptote is by way of an initial overshoot (not visible in the example in Figure 21).

We developed a continuous model (Appendix A) to see what features of the results could be explained analytically. The continuous model predicts the asymptote, but not the oscillations or the overshoot.

We hypothesized that the oscillations are due to phase locking among tasks, based on initial lumpiness in their distribution and the very small variability in their recurrence. To test this, we ran another round of experiments in which tasks require only a single resource, but occur with high variability (standard deviation = 10% of the failure mean). With this configuration, the plots continue to show oscillation and asymptoting, but the initial overshoot is less clear. The hypothesis is not confirmed.

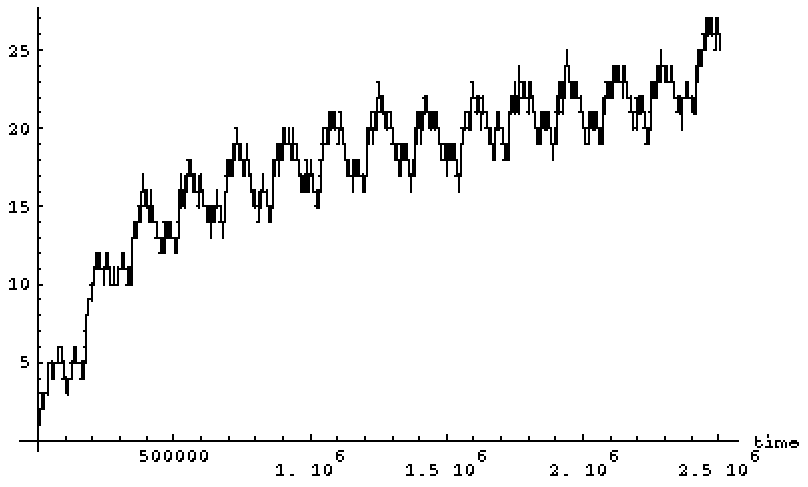


Figure 21 Oscillations in number of tasks waiting

We also explored the effect of

deliberately locking multiple tasks in at the same time with zero variance. With a bucket size of 1, the number of waiting tasks does not change significantly over the run, while with larger bucket sizes, we again get oscillations, asymptotes, and overshoots, but with better-structured oscillations (saw-tooths with rapid rise and slow drop).

We have also done preliminary exploration of a configuration in which the task agents select from among five supplier agents on the basis of a statically weighted preference map. Oscillations are much finer, and asymptote and overshoot are less apparent.

A shift in priorities for the MICANT team led us not to pursue this model further.

4.2 SNAP

The CAMERA system at ISI allocates resources to training missions for Marine aviators. They performed some experiments for us with their own code, and we implemented a version of the RAG, the ISI-RAG, to permit us to conduct further studies.

4.2.1 Problem Structure

Each mission has a number of requirements, each of which can be satisfied by a number of resources. The resources are bombing ranges, pilots requiring training, qualified instructors, and aircraft. The most constrained resource, empirically, is the set of ranges. The CAMERA team has developed a number of resource allocation algorithms. The one currently being transferred to the Marines is a greedy system, which works as follows.

1. Each mission bids first for range time, which is the most constrained resource. It queries the ranges for their available time slots. It receives a list of all the time slots that the range director has made available, including those that have already been committed to other missions. It then marches through all available time slots in 30-minute start intervals looking for a time slot of sufficient length that is not already committed.
2. After securing a range time slot it identifies the pilots requiring that training, qualified instructors, and aircraft qualified for the mission type. It then initiates three parallel requests for commitment (RFC's):
    - The first pilot in the ordered list of pilots
    - All qualified instructors
    - All qualified aircraft
3. From the instructors and aircraft it accepts the first to respond and decommits to all the rest that commit. For the pilot, it will send an RFC to each pilot in turn until it finds the first pilot who can commit.
4. If it cannot find any pilot or aircraft, or instructor to commit to that time slot, it decommits all commitments and samples the next 30-minute start time on the range.

4.2.2 Experiments with the ISI Code

In this system, each task requires multiple types of resources (pilots, instructors, aircraft, and ranges), and resources take the form of time slots. As a convenient way to adjust the relative load and capacity of the system, we hold the number of missions constant at 50, and vary the time horizon over which the missions could be scheduled. In practice the planning horizon is constrained by when the missions need to fly, but for our experimental purposes, we ignore these constraints and focus simply on the percentage of the 50 missions that can successfully be scheduled.

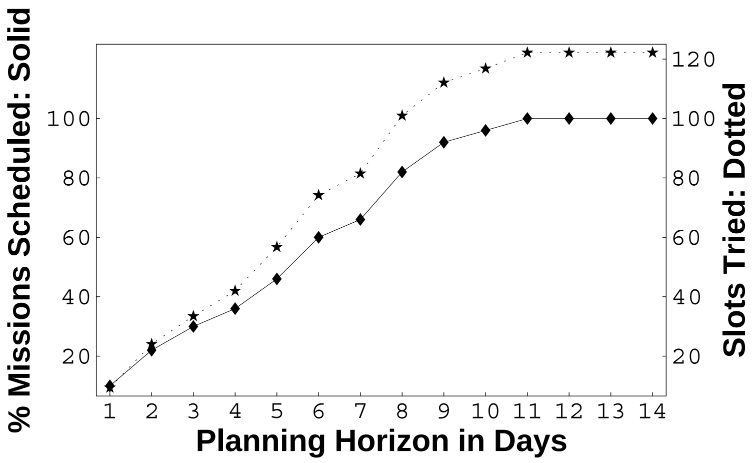


Figure 23: Performance and Effort in CAMERA Baseline.—Both performance and computational load *increase* as constraints *decrease*.

Figure 23 shows the dependency of system performance (% of missions scheduled) and computational effort (the number of slots that have to be tried) on the planning horizon, based on experiments kindly performed for us by Alejandro Bugacov at ISI. The less constrained the system (the longer the horizon), the higher the performance, with no sharp transition. However, unlike the RAG, the effort also increases with the planning horizon. In terms of movement from high to low performance, the profile is neither Easy-Hard (as in some optimization systems) nor Easy-Hard-Easy (as in decision problems and the Mini-RAG), but Hard-Easy! The least work is needed for the shortest planning horizon, when the system is most tightly constrained. The system's greedy algorithm assembles sets of resources for each mission sequentially, and does not explore alternate configurations if the initial configuration fails. Thus the effort expended, like the system performance, is monotonic in system capacity. We speculate that if the missions selected range slots randomly rather than using the greedy approach, there would be fewer collisions among missions vying for the same time slots and the effort would decrease as the planning horizon increased, resulting in an Easy-Hard-Easy profile. These results are discussed further in [24].

4.2.3 ISI-RAG

The ISI-RAG model was developed to account for certain features of CAMERA. Specifically, it addresses issues of assembling a set of specified resources that must align temporally with one another. Thus a Supplier in this model consists of a Resource with a number of Time Slots available. The game proceeds in major and minor cycles. During the major cycles each Task agent attempts to reserve one time slot on the resource. During the *k* minor cycles each Task agent attempts to reserve another slot on the Resource (according to its strategy) until it is successful in finding an open (unreserved) slot. Table 9 lists the entities and parameters for the ISI-RAG.

Table 9: Entities and Parameters for the ISI-RAG

Parameter	Description
N	Number of Task Agents (TAg)
G	Number of Slot Agents (SAg) in resource
S	Number of Slot Strategies (SS) in TAg
L	Number of (equally probable) input states for a SS
k	Length of output sequence of a SS

4.2.3.1 Task Agent (TAg)

Mission.—In each major cycle of the game, each TAg has to acquire one time slot (represented by a SAg) on the resource. The k-sequence in which the TAg approaches the SAg's of the resource is selected by the most wealthy SS of the TAg. The TAg awards its SS according to their performance.

System Knowledge.—A TAg has access to all SAg's within the resource.

Internal Knowledge.—A TAg has S competing Slot Strategies to select the k-sequence SAg's in which it places its orders. A TAg has a certain wealth measured in points gained in the past major cycles.

4.2.3.2 Resource Agent (RAg)

Mission.—In each major cycle of the game, the RAg facilitates the mating of its time slots (SAg's) with TAg's.

System Knowledge.—None.

Internal Knowledge.—A RAg has G time slots, represented by SAg's.

4.2.3.3 Slot Agent (SAg)

Mission.—In each minor cycle until the time slot is taken, the SAg accepts orders from TAg's, selects one winner, and from then on, blocks the time slot on behalf of this TAg.

System Knowledge.—None.

Internal Knowledge.—A SAg manages all the incoming orders in one minor cycle and keeps the reference to the winning TAg.

4.2.3.4 Slot Strategy (SS)

Mission.—In each major cycle, a SS selects a k-sequence of SAg's of the resource, based on the current global state of the system.

System Knowledge.—A SS knows the current global state of the system that is represented by a number from the interval [0,L), which is communicated at the beginning of each major cycle. The SS also knows all SAg's of the resource.

Internal Knowledge.—A SS has a certain wealth measured in points gained in the past major cycles.

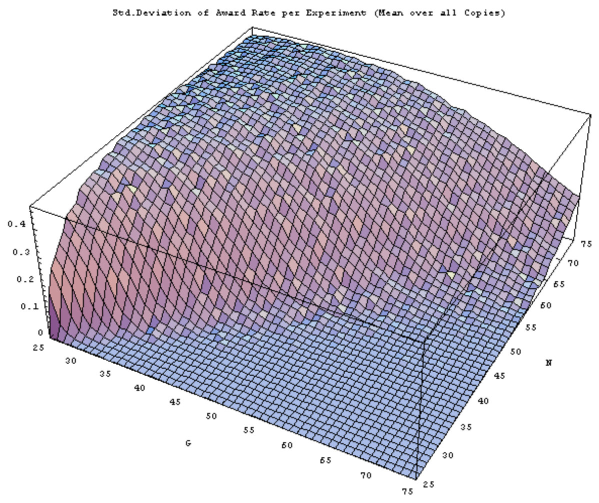


Figure 24: ISI-RAG Mean Award Rate (L=15, K=25, N={25,75}, G={25,75}, S=2)

Figure 24 shows one of the results from these experiments. The Mean Award Rate represents the mean number of points awarded to the agents. Points are awarded based on how quickly the agent's strategy secures it a time slot. The figure shows the standard deviation of this award rate as the number of time slots available (G) and the number of agents looking for a time slot (N) varies. This represents a measure of the behavior of the system. As the standard deviation increases, the agents are less likely to have consistent results. As can be seen from the figure there is a transition in the standard deviation that sharply increases as the number of agents approaches the capacity (number of slots) of the system. This result is consistent with the results we have seen for the mini-RAG and other simpler resource allocation games. This provides further evidence that we should be able to see similar dynamics in the ISI system.

4.3 Marbles

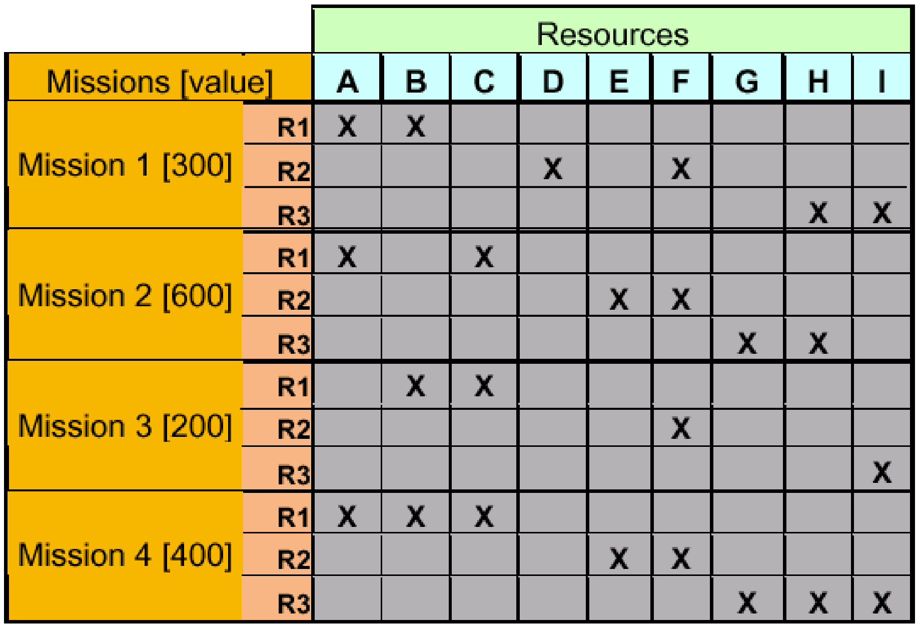
ISI's most mature set of algorithms for scheduling training missions is called "Marbles," because the tasks interact with one another by trading counters much as children trade marbles. Our studies of Marbles have three aspects. We tried, with varying success, to develop a rough estimator of the difficulty of a Marbles problem, then explored the possibility of a Marbles RAG. In the end, our best results came through experiments that we conducted with our own modifications of the Marbles code. These experiments focused on the ISI MarbleSize algorithm, and explored its necessity (i.e., can a simpler solver do as well), and its sufficiency (i.e., can an enhanced solver do better)?

4.3.1 Problem Structure

Figure 25 shows the basic structure of a Marbles problem. There is a set of resources (R) that is available to satisfy a set of requirements (Q). Each requirement is part of one (and only one) task from a set of tasks (T). There also exists a set of eligibility markings (E) (the X's in Figure 25) that denote if a given resource is eligible to satisfy a specific requirement. We call a cell with an X, an "engagement."

Finally, every task is assigned a positive value, and a task value is only 'recovered' if every requirement of the task is assigned a resource. In other words, if a task has all its requirements satisfied, then the total recovered value of the overall problem solution will include the full value of the task, otherwise 0 value is included from the task.

A specific marbles problem is static in the sense that the sets of R, Q, T, and E do not change (i.e., time-invariant). The assignments of resources to requirements may change during the solution generation process,



		Resources								
Missions [value]		A	B	C	D	E	F	G	H	I
Mission 1 [300]	R1	X	X							
	R2				X		X			
	R3								X	X
Mission 2 [600]	R1	X		X						
	R2					X	X			
	R3							X	X	
Mission 3 [200]	R1		X	X						
	R2						X			
	R3									X
Mission 4 [400]	R1	X	X	X						
	R2					X	X			
	R3							X	X	X

Figure 25: The Basic Structure of a Marbles Problem (from [5])

but the R, Q, T, and E sets will remain unchanged.

A ‘valid’ Marbles problem observes the following rules:

- Every resource can fill at least one requirement.
- Every requirement has at least one resource that can fill it
- At most one resource will be assigned to a requirement.
- Every task has at least one requirement.
- Task value is recovered only when all requirements are filled for the task.

The Marbles problems we investigated were in the following ranges:

- R (resources) ∊ [10, 100]
- Q (requirements) ∊ [30, 1600]
- T (tasks) ∊ [5, 400]
- E (eligibles) ∊ [100, 2000]

Some of the analysis provided later looked at smaller problems in order to gain some insight, but simulation experiments worked with the problem scales stated above.

ISI reports on three solver performance measures:

- recovered task value;
- messages sent;
- execution time.

In our experiments, we focused on recovered task values, since some of our comparison methods used centralized algorithms that did not explicitly send messages, and execution time was subject to idiosyncracies in computer configurations.

A	r1	r2	r3	r4	r0
q1	X				X
q2		X			X
q3			X		X
q4				X	X
q0	X	X	X	X	

B	r1	r2	r3	r4	r0
q1	X	X			X
q2		X	X		X
q3			X		X
q4	X			X	X
q0	X	X	X	X	

C	r1	r2	r3	r4	r0
q1	X	X	X	X	X
q2	X	X	X	X	X
q3	X	X	X	X	X
q4	X	X	X	X	X
q0	X	X	X	X	

Figure 26: Sample Marbles Problems

4.3.2 Difficulty Metrics

To guide experimentation, we sought a general characterization of the difficulty of a problem posed in the form of missions, requirements, and resources, as in Figure 25.

Consider, for instance, the simplified problem instances in Figure 26. Each problem has Q =4 and R = 4, and we assume that each requirement is a separate mission, all of equal value. Requirement q0 is a “null” requirement to which an unused resource is assigned, and it is the only requirement that can have more than one resource selected in a valid solution. Resource r0 is similarly a null resource that is assigned to a requirement to which no other resource is assigned, and it is the only resource that can be assigned to more than one requirement.

A ‘Marbles’ problem appears to be most easily solved when the eligibility set (E) is either at its minimum or its maximum size. A metric that captures the size of the eligibility set relative to the other problem dimensions is a density metric, $\delta = E / (R * Q) \in [1/R, 1]$. Intuitively, problem A (at the top of the figure, $\delta = 1/R = 0.25$) should be very easy to solve, because there are no conflicts, either among resources or among requirements. Problem C (at the bottom, $\delta = 1$) should not be much more difficult, since any resource can satisfy any requirement. Problem B ($\delta = 7/16 = 0.44$) should be the most difficult of the three, since there are non-trivial conflicts. For example, if r3 is assigned to q2, no solution is possible for q3. (The notion of complexity as residing midway between two extremes of problem structure reflects the discussion in [11 pp 129-136].]

We explored several approaches to this problem, in addition to ISI's approach. For the sake of this discussion, we assume that the four requirements in the examples in Figure 26 are independent tasks, with values 1, 2, and 3, respectively.

SAT-Based.—ISI's approach to analyzing Marbles problems in the ATTEND project is based on converting the problem to a SAT representation and characterizing it using known characteristics of SAT. We did not pursue this approach for two reasons. First, this effort would have been duplicative of ISI's effort. Second, our philosophy is to try to base our metrics on domain-level characterizations of the problem rather than those particular to a specific computational approach (Figure 3).

Probabilistic Models.—It should be possible to assess the conditional probability of each potential engagement and roll these up into an assessment of the overall difficulty of a given problem. That is, one should be able to replace each "X" with a probability that the given engagement is occupied, such that each row and each column sums to 1. The difficulty of a requirement is the probability that it will not be satisfied (the probability that it will use r0). The difficulty of a task is the sum of its requirement difficulties, and the difficulty of an entire problem is the average of task difficulties, weighted by task values.

We tried several approaches to computing engagement probabilities. All failed either through lack of tractability or because the results they yielded were not intuitively meaningful (e.g., asymmetric probabilities for symmetrically aligned engagements).

- A formal analysis of conditional probabilities, using Bayes' Theorem. This approach proved intractable. In general, the sets of equations generated were overconstrained, (e.g., ten independent equations for seven unknowns) and were nonlinear. Solving such systems amounts to solving the entire problem.
- Focus independently on the columns and the rows.
  - The column (the resource) wants to choose the requirement with the highest value. If the value of requirement *i* is v_i, the resource should assign itself to that requirement with probability $v_i/\sum v_i$, which we call the "myopic value-based probability" (MVP). This initial value automatically satisfies the probability axiom $\sum$MVP = 1 by columns, but does not take into account row interactions. To handle these, we normalize across rows, then across columns, and so forth until the system converges.
  - A symmetrical perspective views the requirement as picking a resource on the basis of a myopic cost-based probability (MCP)

$$p_q = \frac{1/c_q}{\sum_{i \in resources} 1/c_i}$$

where c_i is the cost of resource i. Again, by itself this estimate is inadequate.

- One could naively take the engagement probability as MVP * MCP. This approach yields difficulty scores for (A, B, C) of (0, 0.44, 0.69), which does not satisfy our intuition.

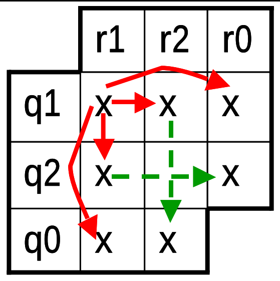


Figure 27: Iterative Structure Extension

- Instead of iterating the values over the entire structure, one could start with a single engagement and iteratively extend it (Figure 27). Initially, one assesses the impact on q1r1 of the engagements in the same row and column (solid arrows). At the next step, one refines this estimate by including the impact of engagements in the rows or columns next further removed (dashed arrows).
- A related approach focuses attention on the graph induced by the problem matrix in the following way: Each engagement is a node, and there is an edge between two nodes just when their engagements share a common resource or a common requirement. Then at the bottom level, the probability of an engagement is 1/(1 + degree of the engagement's node). At the next level, one reduces the contribution of each neighboring node to the engagement's degree, based on the neighboring node's own degree, and so forth. This approach depends sensitively on the connectivity of the graph, and in practice proves to be as complex as actually solving the problem. Even at second order, example B yields a negative difficulty for q4.
- One could also do a Monte Carlo search of possible system states. A state is a vector with as many elements as there are resources. The value of an element indicates the requirement to which that resource is assigned. One generates states randomly, filters out illegitimate states, and then estimates engagement probabilities empirically. This approach is the most promising of those that we explored, but it was the last probability-based approach we considered, and resources did not permit exploring it further.

Ad-Hoc Models.—In the end, the most useful measures were fairly ad-hoc.

A normalized contention metric captures much of the flavor of the previous models, in a more tractable form.

- The contention for an engagement is the number of engagements in its column, divided by the number of engagements in its row. Thus the more requirements want a resource, the higher its contention will be, while the more alternative resources are available, the lower the contention will be.
- The contention for a requirement is the maximum contention across all possible resources, normalized by the sum of the maximum contentions across all requirements.
- The contention for a task is the sum of the normalized contentions of its requirements.

The contention metric for (A, B, C) is (0.250, 0.257, 0.250). Thus it does capture our intuition that case B is more difficult than either A or C.

The only application of the value of a task in the contention metric is in the final weighted combination of the task contentions into an overall problem contention. A more fundamental approach is to ask what proportion of the total value available can be recovered for a given problem. The sum of the task values is an absolute upper bound on the total recovered value from a problem, but a very loose one. For very small problems, it is feasible to generate all possible solutions and find the one with maximum value. The ratio of this maximum value to the sum of task values then becomes a measure of problem ease (and the inverse, a measure of difficulty). Unfortunately the problem is NP-complete, so identification of optimal recovered value is not feasible for larger problems.

As a surrogate, a fairly effective upper bound for an optimal solution can be identified. This upper bound is calculated a follows:

1. Calculate the a value-per-resource required (VPR) for each task by dividing the task value by its number of requirements (and therefore needed resources).
2. Rank all tasks by their VPR from high-to-low.
3. Select tasks for inclusion in the solution upper bound set by choosing the highest VPR tasks until they have used all the available resources. If the last task can only be partially satisfied by the remaining resources, include a pro-rated portion of the task proportional to the available resources.
4. Add up the task values in the upper bound set, and this becomes the upper bound value of a solution.

This estimate is an upper bound because no consideration is given to resolving resource conflicts, and this solution may not be feasible (and usually isn't). In particular, because it ignores E, it gives the same answer (1) for all of our sample problems. Nevertheless, it is a useful metric for evaluating experimental runs, as we shall see in Section 4.3.4 below.

4.3.3 Marbles RAG

We were able to extend the Mini-RAG to capture the characteristics of the Marbles problem. In the Mini-RAG, each consumer has access to all suppliers, and needs a delivery from only one to satisfy its requirements. The Marbles counterpart to a consumer is a task, and the counterpart to a supplier is a resource. There are two important differences between Marbles and the Mini-RAG.

- In the Mini-RAG, every consumer has access to all suppliers, but in Marbles a task has access only to a subset of the resources (those with whom its requirements have engagements).
- In the Mini-RAG, a consumer is satisfied if it gains access to a single supplier, but in Marbles a task is satisfied only if it gains access to as many suppliers as it has requirements.

We model these differences by defining distributions over the number of accessible and required suppliers per customer, and generating Marbles-like configurations accordingly. Preliminary experiments with the Marbles RAG show interesting, non-trivial behavior of mean consumer wealth and the standard deviation of group size with MG's *m* parameter. We have not pursued these dynamics further at this time.

4.3.4 Is MarbleSize Necessary

The upperbound analysis establishes a ceiling on the recovered value for a marbles problem. As we will show, the ISI MarbleSize solver performs quite well in approaching this ceiling. But is it possible that simpler solvers would do as well?

ISI developed a number of other negotiation-based solvers of some complexity that clearly performed worse than MarbleSize. To better understand the range of solver capability, we built a set of 'stupid' solvers. These solvers start with the simplest approach (pure random assignment) and gradually add more problem knowledge to guide their solution strategies. The 'stupid' solvers are required (as ISI solvers are) to observe the five rules outlined in Section 4.3.1. They fall into two classes: resource driven and task driven. Stupid solvers, unless specified, will make random decisions.

Resource-Driven Solvers.—We explored four resource-driven solvers.

- Free Solver: resources are randomly assigned until no more assignments can be made. Tasks may end up with only some requirements filled, and such tasks do not recover their value.
- Holdoff Solver: same as Free Solver, and tasks that reach a point where they cannot be satisfied are not assigned any more resources.
- Recycle Solver: same as Holdoff, and blocked tasks must free resources already obtained.
- Tryagain Solver: same as Recycle, and tasks that are blocked get one last chance (at the end) to get resources.

Task-Driven Solvers.—We explored four task-driven solvers.

- Zipin Solver: tasks are randomly selected to grab all the resources they need.
- Zipinorder Solver: tasks are selected in the order of their value to grab resources.
- Zipinvratio Solver: tasks are selected in the order of VPR to grab resources.
- Zipzip Solver: same as Zipinvratio, and blocked tasks get one last chance (at the end) to get resources.

The 'Zipinvratio' solver uses the same basic approach as the upper bound difficulty calculation, but unlike the upper bound calculation, 'Zipinvratio' must observe the constraints of resource eligibility (E).

We ran the set of stupid solvers, along with the ISI MarbleSize solver, against two different Marbles problems, "Small" and "Medium" (Table 10). Each solver was run 21 times on each problem. Figure 28 plots the results for each solver, ordered left-to-right by recovered value. The upperbound for the problem is drawn as well.

Table 10: Parameters for Test Problems

	Small	Medium
T(asks)	10	100
R(esources)	20	100
Q (requirements)	32	404
E(ligible engagements)	137	3709
Density E/(Q*R)	0.21	0.09
Total Value	16415	28242
Upper Bound Value	11588	17203
Difficulty (Total/Upper Bound)	1.42	1.64

Quite consistently, the ISI MarbleSize solver produces superior results compared to any of

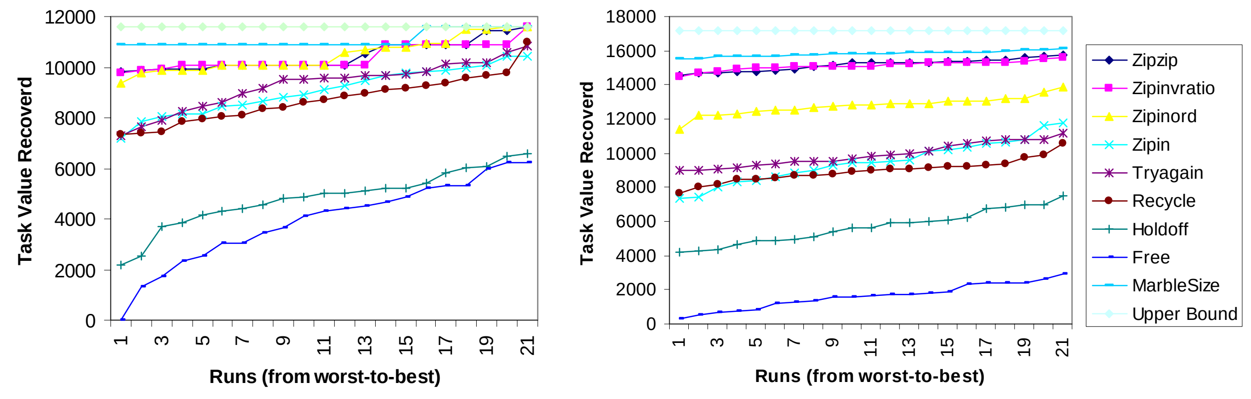


Figure 28: Stupid Solvers on Small (left) and Medium (right) problems.

the stupid solvers.

4.3.5 Is MarbleSize Sufficient?

The ISI MarbleSize solver is clearly superior to simple methods, and comes quite close to the upper bound. The following questions remain:

- Is there a solver that can find a better solution (in terms of recovered value)?
- Is there a solver that can use less messaging to negotiate a solution?
- Is there a solver that has simpler algorithmic complexity?

[12] addresses (at a cursory level) tradeoffs ISI found between recovered value and number of messages. It illustrates clearly that a point of diminishing returns has been reached. Still, a study of the MarbleSize algorithm revealed three potential control points where improvements might be affected:

1. Determination of a MarbleSize (bid value): The MarbleSize solver chooses an initial value based upon a VPR calculation, and modifies it by simple halving. Is there a better way to choose MarbleSize?
2. Determination of a resource set: The MarbleSize solver randomly selects a set of resources to bid on that would satisfy a task's requirements. Is there a better way to identify which resources to bid on?
3. Determination of task dropout: The MarbleSize solver will first try one, then another, or randomly selected resources to bid on. If both sets fail, then MarbleSize will drop the task from further consideration. Is there a better way to determine when to drop out a task?

We focused our attention on the third of these issues, using a variety of the pheromone-inspired control we developed in the Mini-RAG (Section 3.3).

4.3.5.1 Basic Pheromone Mechanism

In nature, a pheromone is a marker used by insects to guide their collective decision processes. Pheromone mechanisms have four components:

- deposit - an entity (insect or simulation agent) marks an event (often spatially) by adding pheromone to an already environmentally existing base of pheromone.
- evaporation - over time, pheromones gradually fade (unless new deposits reinforce them).
- propagation - spatially, pheromones disperse usually with the maximum concentration of a deposit remaining at the original point of deposit.
- sensing - other insects/agents make decisions or take actions based upon the pheromone levels they sense in their environment.

In enhanced MarbleSize, each task agent has exclusive access to its own 'NoBidPheromome' object. (We do not make use of propagation, although it would be interesting to explore the effect of allowing agents with similar VPR's to share pheromone and thus encourage one another to persist or drop out.) Pheromone strength is maintained according to the following equation:

$$\text{pheromoneStrength}(t) = (\text{pheromoneStrength}(t-1) + \sum \text{deposit}(t-1)) * \text{evaporationFactor}$$

where

- evaporationFactor = 0.5, providing for an exponential decrease in the NoBid pheromone strength over time (and a return to task activity) if the circumstances that discouraged bidding previously have gone away
- deposit(t) is a deposit for each bid LOSE message received by a task. The amount of the deposit is limited by a sigmoid function to a value between 0 and 1. The following is the equation for a single deposit:

$$deposit = \frac{1}{1 + e^{\frac{1 - bidave/VPR}{pressure}}}$$

where

- *pressure* represents the amount of contention for a particular resource, and in these experiments is simply the number of bids on the resource;
- *bidave* = (sum of all current bids on resource) / (number of current bids on resource)
- *VPR* is the 'value-per-resource needed' that is computed on a per task basis.

Table 11: Enhancements of MarbleSize

Version	Base Solver	Bid Skipping	Enhanced Dropout	Deadline Response
MarbleSize	MarbleSize	no	no	no
MarbleSize w. hstop	MarbleSize	no	no	hardstop
MarbleSize w. roff	MarbleSize	no	no	rolloff
siggy4-evap.5	MarbleSize	yes	yes	no
siggy4-evap.5 w. roff	MarbleSize	yes	yes	rolloff
quick-siggy4	MarbleSize	no	yes	no
quick-siggy4 w. roff	MarbleSize	no	yes	rolloff

The NoBidPheromone serves its associated task in three capacities.

1. It drives a 'No-Bid' decision.

The pheromone level corresponds to a probability that a task will skip bidding on a given cycle.

2. It drives a 'Dropout' decision. Once the 'NoBidPheromone' level exceeds a threshold, the task altogether drops out from future bidding (and releases already acquired resources).

3. It is used in conjunction with a dynamically decreasing threshold to create a deadline mechanism for finding a solution.

To evaluate the 'NoBidPheromone' enhancements, eleven runs each of 14 different task (and requirement) size configurations were run against an unenhanced (control) version of MarbleSize, along with similar runs against enhanced versions. In all experiments, T = variable, $R = 100$, $Q \sim 4T$, $E \sim 9Q$, Density ~ .09. Table 11 summarizes the versions and their differences. In an additional control experiment set, the deadline was simply a forced stop, or 'hardstop' , of the solver as opposed to a dynamic 'rolloff' of the NoBidPheromone threshold.)

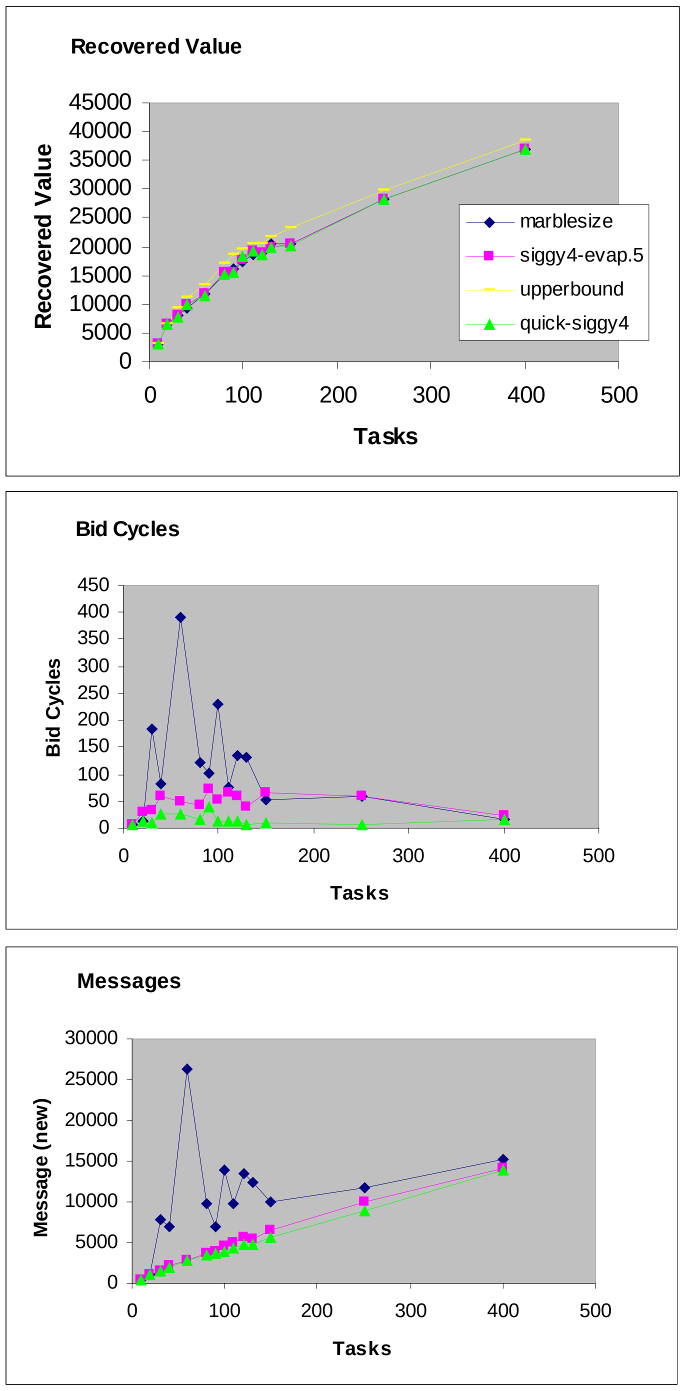


Figure 29: Impact of Pheromone Learning on MarbleSize Algorithm.

4.3.5.2 Non-Deadline Experiments

Figure 29 summarizes the results of the basic (non-deadline) experiments. Each chart compares the performance of the original MarbleSize solver with two enhanced versions. "Siggy" (siggy4-evap.5) uses both the bid skipping and task dropout aspects of the NoBidPheromone. "Quicksiggy" (quick-siggy) uses just the task dropout. We observe:

- All solvers performed at near ideal levels in terms of total recovered value. The maximum possible recovered value, the UpperBound, is drawn as a reference.
- "Siggy" show significant improvement in reducing the number of bid cycles required to solution. "Quicksiggy" did even better.

- Both modified solvers showed significant improvement in message reduction to the point that the number of messages required is approximately linear with the number of tasks (and requirements) in the problem.

Closer investigation leads to the following conclusions based on the above results:

- MarbleSize, or any solver algorithm that uses VPR (value-per-resource required) as the basis of its bidding mechanism, will very likely return excellent solutions in terms of total recovered value. The basic MarbleSize solver sets the initial bids (the "MarbleSize") for all agents to be exactly the VPR. (Note: the upperbound for total recovered value is calculated by using the VPR.)
- Most final task-to-resource assignments are achieved on the first 2 cycles.
- Use of the 'NoBidPheromone' results in quickly identifying less desirable tasks and removing them from the bidding processing. By doing so, both messaging and cycle time are significantly reduced with little or no loss in quality of solution (i.e., total recovered value).

4.3.5.3 Deadline Experiments

A second set of experiments (Figure 30) explored the introduction of a deadline into the solver process, requiring that a solution be delivered in a fixed amount of cycles. The above graphs compare quality of solution for each solver when a deadline is imposed. 'MarbleSize with hardstop' is simply the regular MarbleSize algorithm with a simple cutoff applied. The other versions use the 'NoBidPheromone' in conjunction with a rolling-off threshold:

$$Threshold = Min\left(\sqrt{Deadline - t}/2, 1\right)$$

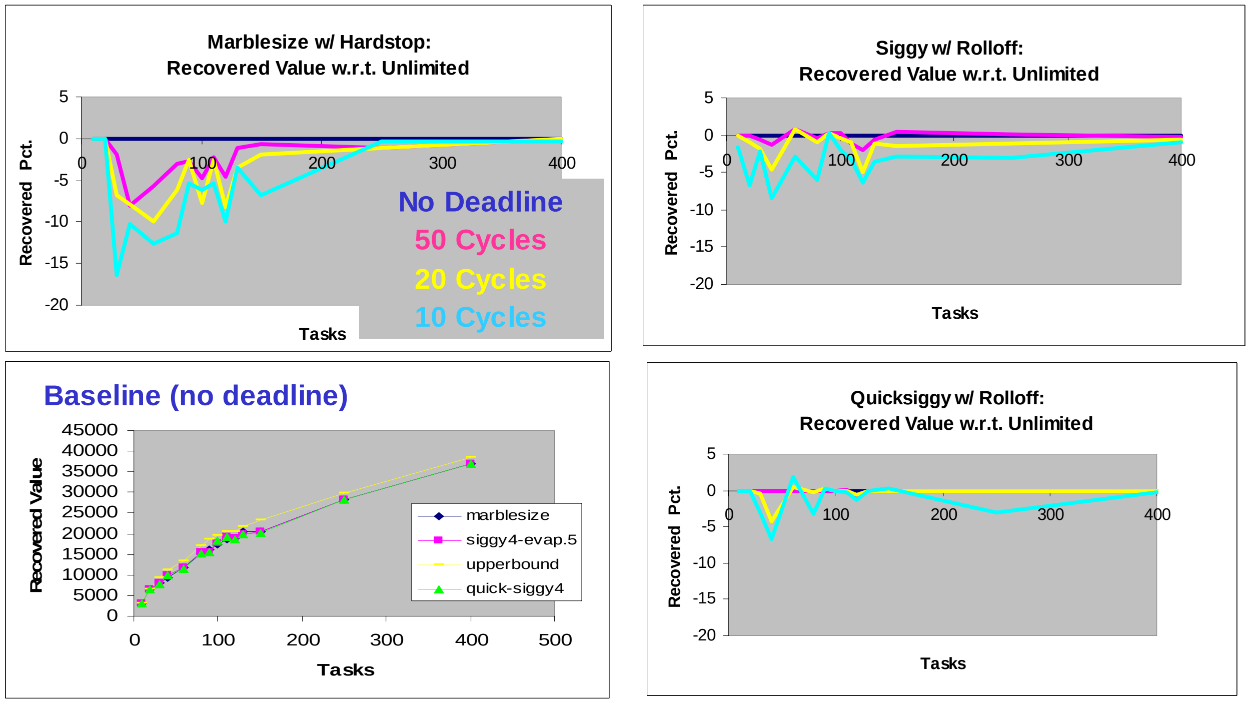


Figure 30: Impact of Deadline Mechanism on MarbleSize Algorithm

As the solver approaches the deadline, the threshold for task dropout decreases making dropout more likely. At the deadline, the threshold is zero, making dropout a certainty for all unsatisfied tasks. The data show that solvers that have already made task dropout decisions early, such as "Siggy" and "Quicksiggy," are least sensitive to the application of a deadline.

5 ECM Tracks

In the ECM problem domain, we worked with both the University of Kansas and Kestrel.

5.1 University of Kansas Track

At Lake Tahoe, the University of Kansas team reported a nonlinearity in effort as an effect of the amount of information transferred, reminiscent of the behavior in MG (cf. Figure 2). We worked with them to explore this phenomenon. Since the UK system was tightly integrated with the RadSim framework and did not support faster-than-real-time simulation runs, UK prepared stripped-down version of their Case-Based Reflective Negotiation Model (CBRNM) framework for our experimentation.

Agents in the CBRNM base negotiations on their past experience, stored in a case base. In the application we studied, two agents negotiate over access to CPU time. The initiating agent keeps sending facts (evidence) to convince the responding agent that it should give up some CPU usage. The requested amount is defined by the parameter *CPUShortage*. The responding agent weighs each fact and adds the weight to its internal evidence level, which represents the amount of CPU the responding agent is willing to give up. If this amount becomes larger than the requested amount, the responding agent agrees to give up the requested resources, up to its *maxCPUgiveUp* threshold. If the initiating agent requests more than *maxCPUgiveUp*, the responder makes a counter offer. If the negotiation runs over the allotted time, it is aborted and no resource is exchanged.

In these experiments, all negotiations run to completion (no timeout). The *maxCPUgiveUp* threshold of the receiving agent is 10%, so all negotiations for less than this threshold succeed. But the effort (number of messages exchanged) increases as the amount of CPU requested moves closer to the threshold, since it takes more evidence to convince the responding agent. Above the threshold, the number of messages required to convince the responding agent is constant, because the sequence of facts does not vary in this experiment. We define the success rate as the ratio of the granted resource and the requested resource. Above *maxCPUgiveUp*, the expected success rate is *maxCPUgiveUp*/*cpuShortage*. For all settings of the *cpuShortage* parameter, there is enough time and enough facts available to convince the responding agent to give up the requested resource. We never reach a timeout failure and we never see a counter offer caused by insufficient evidence.

Figure 23 shows performance and effort in

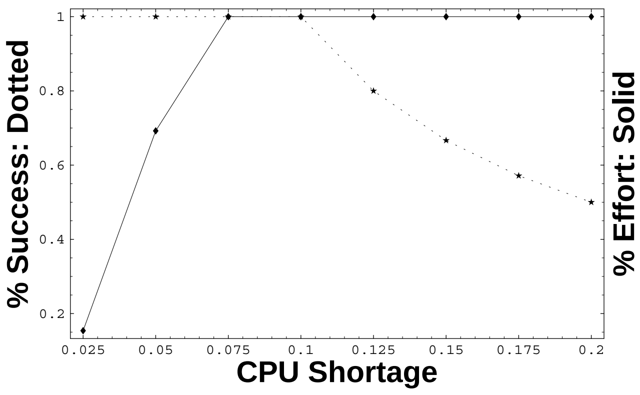


Figure 31: Performance and Effort in Case-Based Negotiation.—Number of messages increases, and success de-creases, as *MaxCPUGiveUp* parameter increases.

this simple scenario. From the structure of the problem, the distinguished point where one might expect phase-transition-like behavior is when *cpuShortage* = *maxCPUgiveUp* (0.1). Though there is no sharp transition, as load increases with respect to capacity, the quality of solutions decreases, and the amount of computation needed (measured by messages exchanged) increases.

Further investigation of this problem showed that some of the abstractions made when UK simplified their system for our experimentation had the effect of precluding more complex behavior, and we did not pursue this model further.

5.2 Kestrel

Kestrel recommended the graph coloring problem as an appropriate abstraction for their approach to the ECM challenge problem. We constructed a variety of RAG, the ColorRAG, to explore the dynamics of this system.

5.2.1 Algorithm

The Kestrel algorithm starts with a random graph of *N* nodes connected to *K* neighbors via unidirectional edges. The number of nodes and neighbors does not change over time. We have implemented six different graph construction mechanisms, summarized in Table 12. All results reported here are from random graphs generated by the Minimum Neighbors algorithm. Measurements (not reported here) show that these graphs are characterized by a short characteristic path length (approx. 1.3) and medium clustering coefficient (approx. 0.68).

At any point in time, each node is assigned one of *G* colors. The assignment of colors to nodes may change over time. A node is capable of perceiving the current color of its neighbors. A

Table 12: Graph Construction Algorithms.

2D Field	The nodes are randomly distributed on a 2D square with a side of $\sqrt{\pi N/(K+1)}$. On such a square, a radius of length 1 about a node includes on average *K* other nodes. For large N this algorithm produces a mean number of neighbors close to K. For smaller N the characteristic neighborhood size varies widely.
2D Lattice	The nodes are distributed deterministically on a square lattice with a spacing of 1. Any node within a fixed radius is counted as a neighbor.
Minimum Neighbors	Beginning with *N* nodes with no edges, we repeatedly pick a node randomly from the set of nodes with the least number of neighbors. For this node we successively add neighbors until there are K neighbors. Any neighbor is again selected randomly from the set of nodes with the least number of neighbors. The process continues until fewer than two nodes with fewer than K neighbors remain. The algorithm tends to produce tightly connected graphs.
Edge Probability	All possible unordered pairings of nodes are considered. The nodes of a pairing are connected with a probability of K/N. Thus, the average node ends up with K neighbors.
2D Nearest Neighbor	The nodes are randomly distributed on a 2D 1 by 1square. In canonical order, we connect each node to its nearest neighbor on the square until it has K neighbors. The resulting graph may not necessarily be planar, but violations will be local.
3D Nearest Neighbor	The nodes are randomly distributed in a 3D 1 by 1 by 1 cube. In canonical order, we connect each node to its nearest neighbor on the square until it has K neighbors. The resulting graph may not necessarily be embeddable in three dimensions, but violations will be local.

change in a neighbor's color is perceived after a delay of CL (communication latency) time units. All nodes share a global probability value AL (activation level), which determines each node’s probability to activate its local reasoning mechanism. If activated, a node re-evaluates its color assignment based on the local Degree of Conflict (DoC, the number of neighbors that share the node’s color divided by the overall number of neighbors *K*). The node calculates DoC for each of the *G* possible colors, using the perceived color of its neighbors and compares the resulting DoC values with the DoC of its current color. Any color whose DoC fulfills a *movement direction* constraint (determined by the algorithm used), is placed into a set of available colors. The new color of the node is selected from this set of available colors based on a *color selection* rule. There are three movement direction constraints and three color selection rules that can be combined to form 9 unique algorithms as described in Table 13.

At the central configuration, activation level AL = 33%, probability of node failure RP = 0% and communication latency CL = 1 cycle. The central configuration comprises 44 nodes (*N*) with 30

Table 13: Color Change Algorithms

		First Decision: Movement Direction (MD)		
		Any	Lateral and Up	Only Up
Second Decision: Color Selection (CS)	Random	• any color may be selected • each color has equal probability (P[C_i]=1/*G*) to be selected *Random walk through color space independent of any other node*	• any color may be selected • each color has equal probability (P[Ci]=1/G) to be selected *Random walk through color space independent of any other node*	• only colors with smaller DoC than the current DoC may be selected • each color has equal probability (P[Ci]=1/G) to be selected *Strict (no lateral drift) random hill-climbing algorithm on the local DoC*
	Roulette	• any color may be selected • the probability of an available color to be selected is inverse proportional to its DoC *Unconstrained probabilistic hill-climbing algorithm on the local DoC*	• any color with equal or smaller DoC than the current DoC may be selected • the probability of an available color to be selected is inverse proportional to its DoC *Relaxed (with lateral drift) probabilistic hill-climbing algorithm on the local DoC*	• only colors with smaller DoC than the current DoC may be selected • the probability of an available color to be selected is inverse proportional to its DoC *Strict (no lateral drift) probabilistic hill-climbing algorithm on the local DoC*
	Best	• any color may be selected • the avaliable color with the smallest DoC is selected *Unconstrained deterministic hill-climbing algorithm on the local DoC*	• any color with equal or smaller DoC than the current DoC may be selected • the avaliable color with the smallest DoC is selected *Relaxed (with lateral drift) deterministic hill-climbing algorithm on the local DoC*	• only colors with smaller DoC than the current DoC may be selected • the avaliable color with the smallest DoC is selected *Strict (no lateral drift) deterministic hill-climbing algorithm on the local DoC*

neighbors each (K) and 8 possible colors (G). The nodes consider any color (MD = Any) but select randomly among only those that give the lowest local degree of conflict (CS = Best). We execute 100 replicas of any chosen configuration (with different random seeds) and we collect the report data between system cycle 9,500 and 10,000 (samples of each node's activation choice).

Table 14: Correspondences between Mini-RAG and ColorRAG

Characteristic	Mini-RAG	ColorRAG
Number of entities that need to obtain a resource	*N* consumers need access to a supplier	*K* nodes need access to a locally distinct color
Number of options from which a consumer chooses	*G* suppliers	*G* colors
Decision strategy	*S*, *m*	MD, CS
Capacity of suppliers	*C*; overload if $N > C$	Overload if $K > G$

Table 14 identifies key points of correspondence between the ColorRAG and the MiniRAG.

Table 15 lists the model parameters of the central configuration. We explore the parameter space around this configuration.

5.2.2 Metrics

We studied four metrics: Global Degree of Conflict (DOC), Option Set Entropy (OSE), False Information Percentage (FIP), and Time to Solution (TTS) (this last only in studying deadline behavior). In each case we plot the mean over the reported cycles.

Global Degree of Conflict (DOC).—The number of edges in the graph that connect two nodes with the same color divided by the overall number of edges. The metric directly measures the global performance of the system, since a DoC of zero indicates a perfectly colored graph. We usually plot the metric multiplied with the number of available colors, since then a value of one matches the expected outcome

Table 15: Central Configuration

Model Parameter	Value(s)	Description
Activation Level (AL)	33%	probability of a node to be activated in a cycle
Reset Probability (RP)	0%	probability of a node to fail (randomly select color) in a cycle
Communication Latency (CL)	1	number of cycles before a color change becomes visible to neighbors
Population Size (*N*)	44	number of nodes in the graph
Neighborhood (*K*)	30	number of edges leaving each node
Color Set (*G*)	8	number of colors available to a node
Graph Construction (GC)	Minimum Neighbors	algorithm connecting nodes
Movement Direction (MD)	Any	restriction on the change in local Degree of Conflict metric
Color Selection (CS)	Best	method to select next color in permitted movement direction
replicas	100	repetitions (vary rnd seed) of the execution of a model configuration
reporting cycles	9500-10000	cycles in which reporting data is collected

of perfectly random choices by all agents.

Option Set Entropy (OSE).—For each node, the option set entropy is the entropy over the probability that a particular color is chosen in a cycle, normalized by the maximum entropy that is given in the random choice among all G colors. For the particular decision mechanism selected in this set of experiments (MD=Any, CS=Best), all colors that do not result in the lowest local degree of conflict in a cycle, the selection probability is zero. All remaining colors share the same non-zero probability, which add up to one. The maximum entropy is given as ln(G). The metric reflects the level of guidance that each node has in its local decision process, since an OSE of zero indicates that there is only one option available, while an OSE of one occurs with a random choice across all colors.

False Information Percentage (FIP).—For each node, the false information percentage is the portion of neighbors, for which the current color assumed in the node's decision process is not the actual color of the node. This difference reflects the impact of the communication latency in the model, since a change in a neighbor's color requires some time (here always one cycle) to become known to a node. The metric reflects the impact of the physical reality of restricted information exchange on the decision processes of the nodes. In the case that FIP is zero, a node has perfect knowledge of the state of the environment, while a value of one indicates that all input to the local optimization function is misleading.

Time to Solution (TTS).—Experiments on temporal aspects of problem solution (e.g., deadline performance) require an additional temporal metric that estimates how long it takes the system to asymptote to a solution. An experiment explores a set of configurations (e.g., N vs. AL). We execute multiple runs (e.g., 64) at each configuration, and run each configuration long enough to asymptote safely, capturing the time series of metrics (e.g., DOC) for each run. Then we construct the Mean Time Series (MTS) over the runs for each configuration, compute the Delta between the first point in each MTS and the Stable Point (the mean over the last 50 points), and define the Threshold as 2% of the maximum Delta over all configurations. In the MTS for each configuration, we measure the distance of each point from the Stable Point, and declare that the solution has been reached if this distance is less than the Threshold.

5.2.3 Phase Transitions

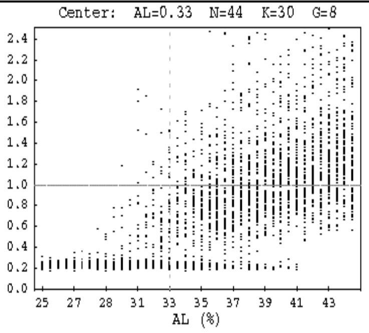
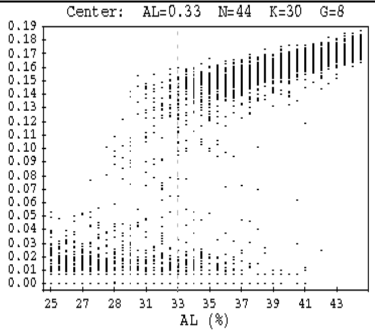
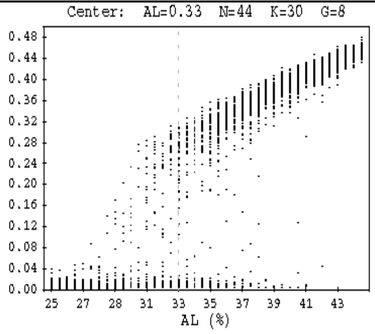
We performed a set of experiments in which we observed the occurrence of a phase shift (with overlap region) in the ColorRAG model. We observed the phase shift varying several parameters around a central configuration while collecting data on three metrics.

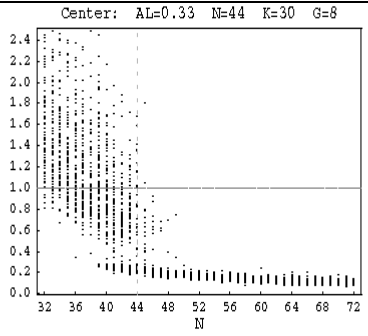
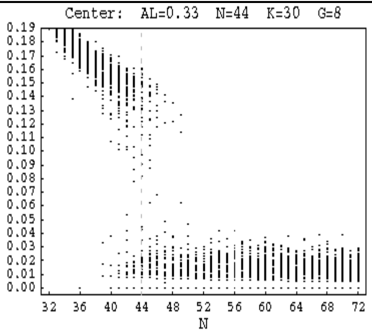
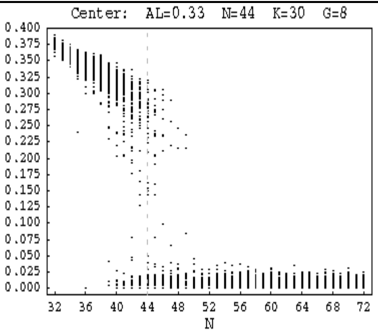
To show the phase change in the dynamics of the system we vary four parameters independently around the central configuration as shown in Table 16 and plot our three metrics for each individual replica:

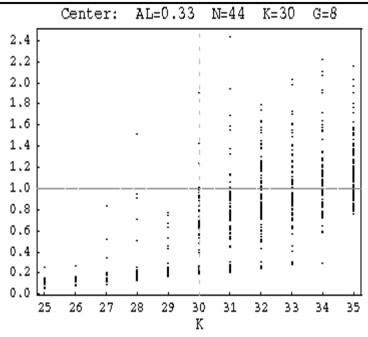
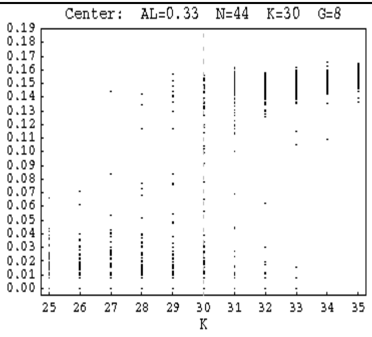
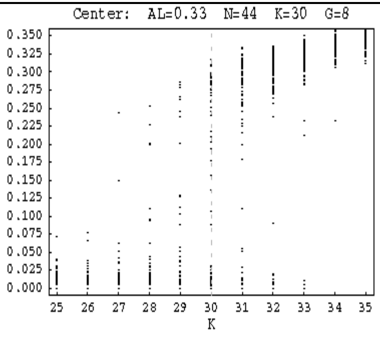
In the following, we discuss the observed changes in the system dynamics as we vary these parameters.

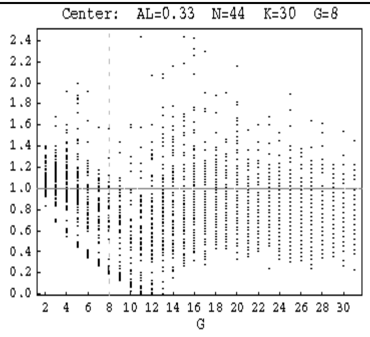
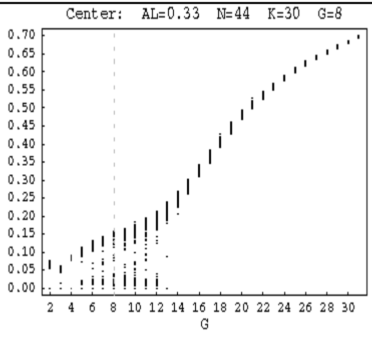
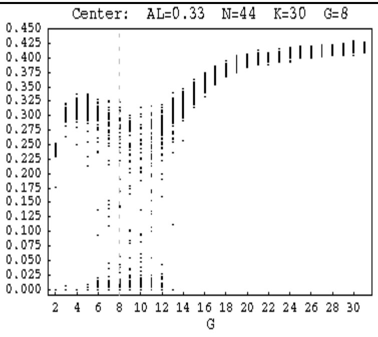
Table 16: Parameter Sweeps for Studying Phase Transitions

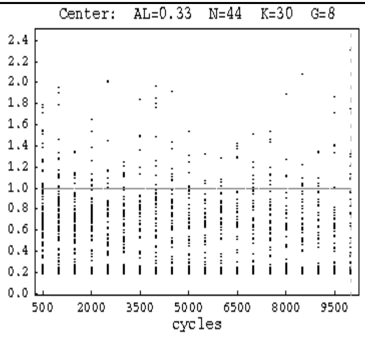
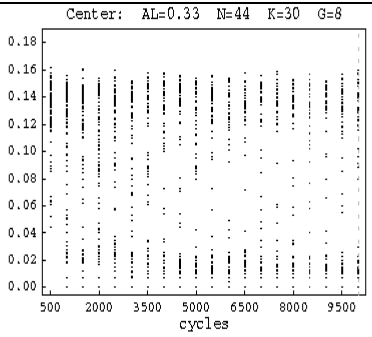
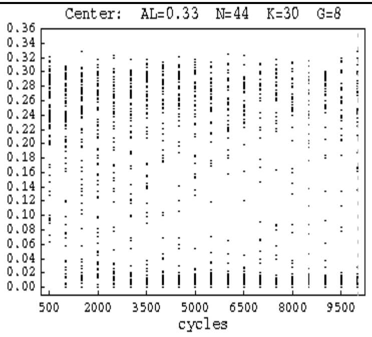
Parameter	First Value	Last Value	Step Length
AL	25%	45%	0.5%
N	32	72	1
K	25	35	1
G	2	31	1
cycles	0-0.5k	9.5k-10k	0.5k

AL = [25% , 45%] by 0.5%		
DoC	OSE	FIP
		
Low AL (<30%) produces a tight band of well performing systems (DOC * G approx. 0.25). Medium AL levels (30% - 38%) correspond to an overlap region, where we still see some systems reaching the well performing region, while (as we increase AL) more and more systems show much worse performance in a wide scatter around the performance expected from a random system. Above an activation level of approximately 40% nearly no well performing systems are observed.	In well performing systems, which are associated with low activation levels, individual nodes get good guidance for their color selection. Given the chosen local decision process (MD=Any, CS=Best), we can infer that there are only a few colors for any node to choose from. In fact, we observe a number of systems that reach an OSE of 0.0 - an indication that no node can choose any color but the one that it already carries (one option). In other well performing systems there can be only a few nodes left that still could change their color. As we move to higher AL, there is a dramatic change in the dynamics of the system. More and more systems fall into the phase, where the average node has multiple colors to choose from randomly (all would result in the lowest local DOC).	Lower AL automatically reduces the false information percentage since the probability of using outdated information reduces as the nodes activate less often. Also, a system in which the nodes change their color less often should produce lower FIP values. This FIP plot shows the combination of these two affects as systems with lower AL values are already settled into their colors and thus a speedy information exchange is less critical.

N = [32 , 72] by 1		
DoC	OSE	FIP
		
Increasing *N* while keeping K fixed reduces the connectedness of the overall system (increase in path length). Systems with few nodes (<40) show a wide range of bad performance. Almost completely interconnected systems (*N* near *K*) perform worse than random - an indicator for thrashing. Between 40 and 46 nodes, the opportunity for significantly improved performance arises as a tight band of good solution is established. This band eventually dominates the system dynamics and above 47 nodes, no results outside the DOC**G*=0.2 band are observed.	As with the experiments that varied the nodes' activation level, the change to well performing systems is reflected in the drastic reduction in the option set entropy of the average node in the system, with some systems completely settled into a local minimum (OSE=0).	While the changes in the false information percentage metric in the case of varying activation levels were driven by two processes, in the varying-N experiments only the effect of the system settling into a good configuration is reflected in this plot. As the nodes have few options to choose from, color changes become less likely and thus the impact of the communication latency on the decision processes is reduced.

K = [25 , 35] by 1		
DoC	OSE	FIP
		
Increasing *K* has the same effect as decreasing *N*; it reduces the characteristic path length of the system and thus increases the global connectedness. Thus, just as in the varying-N experiments, the performance of the system decreases with decreasing path length (increasing K).	*K* > 31 prevents the system from reaching a (mostly) stable point and thus nodes remain able to choose randomly among multiple colors.	The transition from stabilized systems (K<27) to unstable systems (K>31) results in increased color changes with the resulting increased probability of false information used in local decisions induced by the communication latency.

G = [2 , 31] by 1		
DoC	OSE	FIP
		
A change in *G* should directly influence the system's ability to reduce conflicts between neighboring nodes. And in fact, as we increase G from 2 to about 12 colors, we see a tight band of increasingly well performing systems, which eventually reaches perfectly colored graphs (DOC=0). Remarkably, in this range (2 <= G <= 12) other systems show a wide spread of significantly worse performance, even though their overall performance also increases with increasing G. For *G* > 12, the dynamics of the system changes dramatically. Suddenly, not only the band of well performing systems vanishes, but also the degree of conflict of the not so well performing systems increases visibly. For much larger G, a wide band of results stabilizes that extends more below than above the expected performance of random systems (DOC*G=1).	OSE highlights the observed phase change. The band of well performing systems at lower values of G generally corresponds to systems that have already settled down at configurations that do not allow the nodes to choose among multiple colors. The not so well performing systems above this tight band are still unsettled, but, as we increase G, their OSE only grows moderately. After the change of phase at G=12, the rate of OSE increase with G is much higher taking the dynamics quickly towards highly random choices across the full color spectrum. There the nodes have only few guidance in their decision since the set of colors that promise the lowest local degree of conflict dramatically increases in size.	Again the changes in the FIP metric are influenced by two factors. Dominating the plot is the difference between settled systems (lower band) and systems with high rates of color change (upper band). But, especially shaping the upper band is the fact that lower values of G reduce a node's options for change and thus reduce the induced errors in other nodes' decision input. With large numbers of colors available to choose (see high OSE values), the probability for an actual color change increases and thus the impact of the communication latency increases (driving up FIP).

Mean over cycles = [0-0.5k , 9.5k-10k] by 0.5k		
DoC	OSE	FIP
		
These plots sample our three metrics at different times in the life of the system. Previous experiments show that the systems generally settle on their respective dynamics (fixed patterns, drifting, thrashing, ...) within approximately 500 cycles. Our other experiments in this series, in which we vary model parameters, all collect data between cycle 9500 and 10000 to ensure representative results. As we can see in the time-varying plots, the system dynamics are really established early on and do not change significantly. We expect that larger systems (many more nodes) take longer to find their respective dynamics. Also, with the potential of inhomogeneities in the graph structure, sub-regions with different stable dynamics could be established.		

The overall story emerging from this analysis is that phase shifts are ubiquitous in the Color-RAG.

5.2.4 Dynamic Attractors

Sometimes a dynamical system will exhibit different forms of behavior over time, as the system wanders into the basin of attraction of different dynamical attractors. An example of this for the Color RAG happens in the system where GC = 2D nearest neighbor, CL = 1, $N = 600$, $K = 150$, $G = 4$. Figure 32 shows the dependency on AL. The dashed line marks a phase transition such as those discussed in Section 5.2.3. For even higher AL, between 0.4 and 0.7, individual experimental instances fall into one of two bands.

Further observation shows that these bands represent regions of attraction, that every experiment begins in the upper band, and then falls into the lower band if one waits long enough. Qualitatively, the system's behavior in these two bands is quite different. Systems in the upper band color subregions of the graph the same, but differ from region to region (Figure 33a), and the coloring changes drastically from one cycle to the next. Once the system transitions to the lower band, the various colors are distributed more evenly over the whole graph (Figure 33b). A small subset of nodes lock onto one color that gives them a low DOC (the small dark blue nodes in Figure 33b), while the other nodes cycle through the other colors, one or two at a time.

The time required for an instance to transition from the upper to the lower band varies widely. Some instances make the transition in 200 cycles; others take longer than 2000 cycles. Understanding whether a given system will transition quickly or slowly is clearly an important subject for ongoing research. The set of stable, low-DOC nodes that emerges in the lower band could be critical for achieving reliable system performance, and being able to predict how long a particular system instance will take to reach this attractor is essential in deploying operational systems. We hypothesize that the difference is related to the detailed structure of the graph, which is instantiated randomly for each experiment. Testing this hypothesis requires further research.

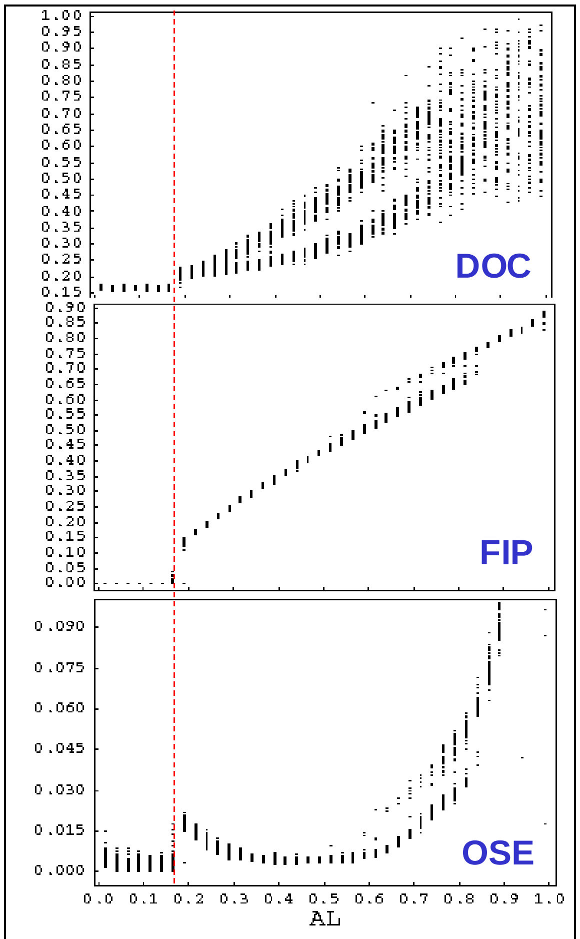


Figure 32: Phase Shift and Attractors

5.2.5 *Parallel with MG*

The investigations of the ColorRAG described up to this point focus on the three non-temporal metrics DOC, OSE, and FIP. When we explore the variation of TTS (Time To Solution) over the parameter space, further interesting structure emerges that shows a high-level correspondence with the dynamics of MG. Figure 34 shows the TTS landscape as a function of AL and *G*.

Consider first the dependency of TTS on AL. Figure 34 shows several basic features as AL increases. Colored dots superimposed on the TTS surface show the *G**DOC, which is 1 under random choice, < 1 for DOC better than random, and > 1 for DOC worse than random.

- For very low AL, TTS is very high, since the system can take only very small steps toward solution. However, in a stable environment (RP = 0, as here), eventually the system reaches a good solution.
- Up to a point, TTS decreases with increasing AL, as the system takes larger steps toward the solution.
- The red line indicates the location of a phase shift. At this point TTS is increasing with AL, indicating the onset of thrashing behavior, as discussed in the previous section.
- This thrashing leads to a peak in TTS, which then

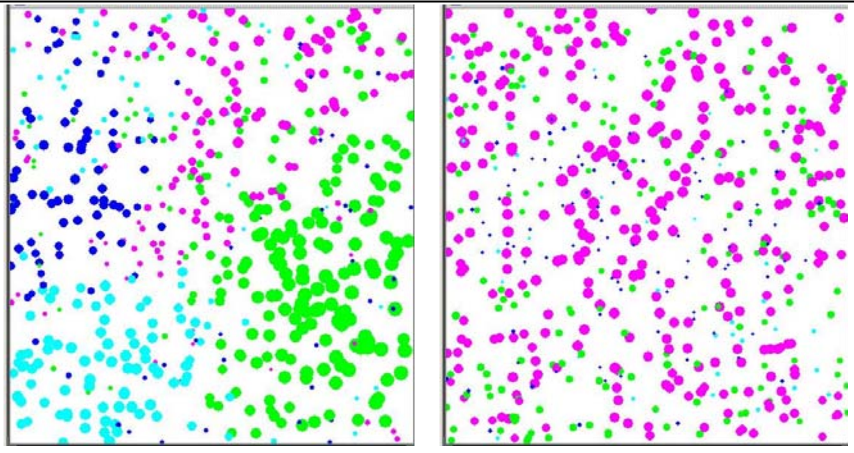


Figure 33: Coloring Patterns in Different Attractors.—Node size is proportional to DOC.

drops to low solution times with poor results.

If AL is too slow, nodes cannot keep up with dynamic changes. But if AL is too fast with respect to CL, nodes make decisions with obsolete information and thrash. Very high AL produces essentially random behavior. This observation is critical in the light of contemporary tendencies to reducing the cycle time in information systems (essentially, increasing AL). Such increases are useful only up to a point, beyond which they introduce instability. Our result here is comparable to the behavior observed in [13] in a related domain.

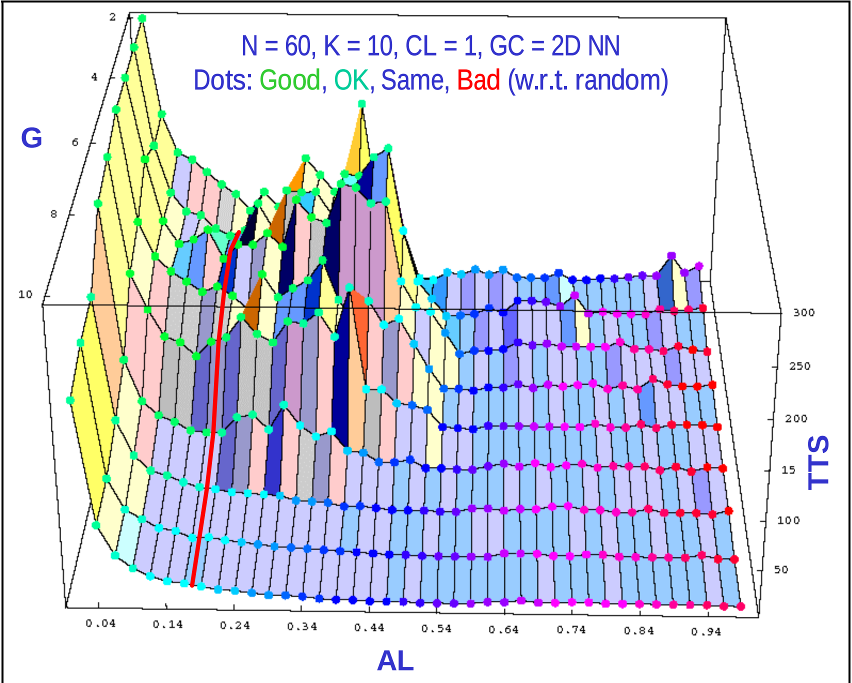


Figure 34: Time To Solution as function of Activation Level

Now consider the dependence on *G*. Where *G* is low, there is no potential for solution, and random works fine (yielding poor results). This region is on the far side of Figure 34. Medium *G* requires effort to find a good solution. For high *G*, random again works as well as anything else, this time yielding trivially good solutions.

Thus throughout most of the parameter space, the system quickly converges, either to very poor values (high AL) or to good ones in an underconstrained system (high *G*). The interesting region is in the corner with low *G* and low AL. The variation in this region is a saddle point, characterized by $\partial^2 TTS / \partial AL^2 > 0$ but $\partial^2 TTS / \partial G^2 < 0$. Such a structure is an important feature in dynamic

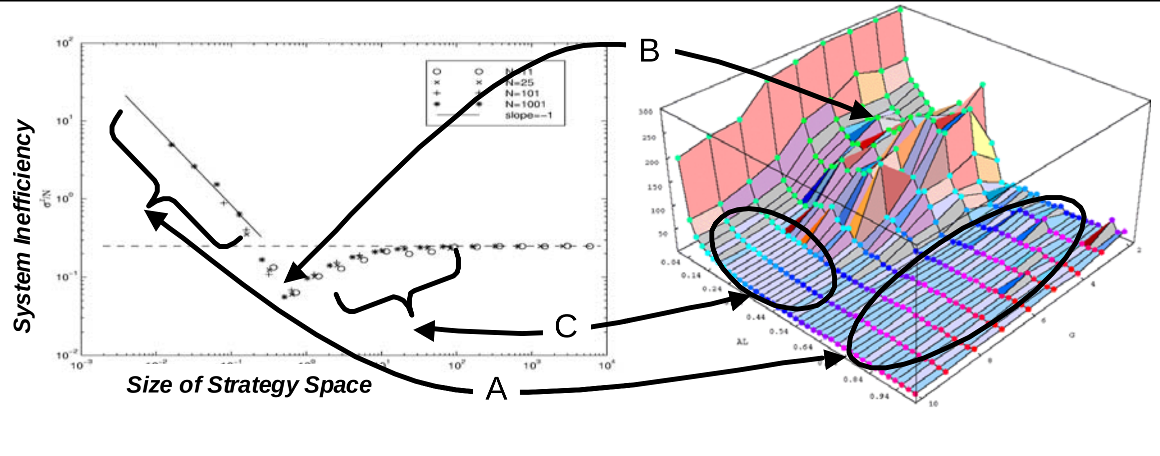


Figure 35: Comparison of ColorRAG and MG

systems.

Comparison of Figure 34 and Figure 2 shows a striking correspondence (Figure 35) that is an example of our Generality Hypothesis.

- In the A regions of both systems, agents have few choices. Their interactions lead to systematically wrong information and bad solutions.
- In the B regions, agents have a manageable number of choices, and the right information is available to enable them to select good solutions among them.
- In the C regions, agents face a surfeit of information and more choices than they can manage, and solutions are neither as bad as in region A nor as good as in region B.

5.2.6 Control Mechanisms

Figure 34 suggests that an effective means of control is adjusting AL. Performance is worst in the high AL region, so our basic control mechanism will be to reduce AL. Even when AL says that a node should evaluate its color assignment, it will not evaluate with probability NAP (No-Action Probability). This probability is set using a variety of the pheromone learning we applied in the Adaptive Mini-RAG (Section 3.3). We have explored two influences on the size of the deposit.

1. The deposit might increase with FIP, since FIP is diagnostic of system thrashing.
2. The deposit might *decrease* as OSE *increases*, since OSE indicates system convergence.

Figure 36 shows the impact of these mechanisms on both quality of solution (*G**DOC) and TTS. Both mechanisms increase TTS (lower row of plots), since the NAP effectively reduces AL and

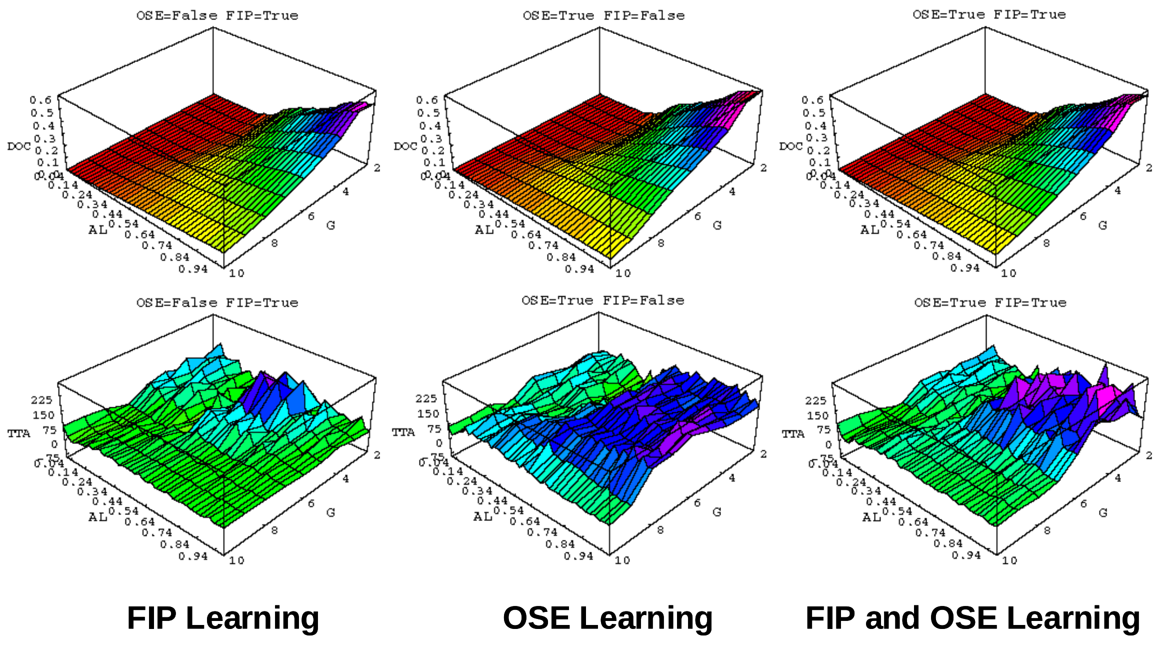


Figure 36: Control Mechanisms.—The upper row of figures shows the improvement in *G**DOC (higher is better) under each control mechanism. The lower row shows the change in TTS (lower is better).

thus the system require more cycles to converge. OSE learning provides greater increase in solution quality than does FIP learning, but greatly increases TTS in the high-AL and high-*G* regime. Combining the two mechanisms (right-hand plots) yields an increase in quality as good as that achieved with OSE alone, but restricts the increase in TTS to the region where the increase in quality is actually achieved.

To make our control sensitive to deadline constraints, we focus on the two-step nature of the color choice algorithm (Table 13): first select a *movement direction*, then invoke a *color selection* rule. The color selection rule has available only the set of colors identified in the movement direction step.

Most of our ColorRAG experiments use MD = Any (allow a node to choose any of the available colors, even those that would increase DOC) and CS = Best (pick randomly among the colors that would yield the lowest DOC). We apply deadline control by modifying the MD decision. With Only-Up Probability *OUP* = $(e^{a*t}/deadline - 1)/(e^{a} - 1)$, where *a* is a tuning constant, we apply MD = OnlyUp instead of MD = Any. The difference between the two is subtle but important. MD = Any with CS = Best means that if there are other colors that would yield the same DOC as the current color, the node can select them, and thus move laterally in search space, *exploring* new configurations. MD = OnlyUp means that the set of colors on which CS operates includes the node's current color and any colors that would yield a *lower* DOC, but no colors that would yield the *same* DOC as the current color. Thus the more frequently a node chooses MD = OnlyUp, the more rigorously it hill-climbs, *exploiting* the gradient it sees.

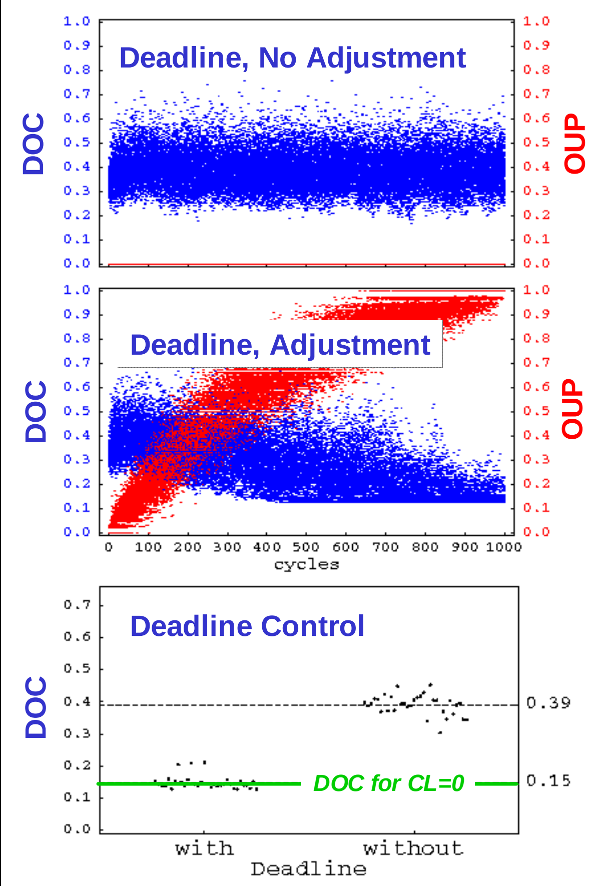


Figure 37: Deadline Control in the ColorRAG.—Application of deadline control allows the system to reach a degree of conflict as low as the best results achieved with perfect knowledge (CL = 0).

Figure 37 shows the result of this mechanism applied to a system with deadline = 1000. The two top plots show the evolution of DOC and OUP as the system runs. Without control, OUP remains at 0 and DOC quickly stabilizes around 0.39. With control, as OUP grows, DOC drops. The bottom plot shows that different runs with deadline control cluster around the level of the best DOC achievable with perfect knowledge, while uncontrolled results have much higher DOC.

Another way to look at Figure 37 is to observe that the behavior seen in the upper plot would also result if we had no deadline at all. That is, the use of OUP control does not just enable us to cope with a deadline. It actually improves the behavior of the system over the case where there is no deadline, by forcing the system to shift from exploration to exploitation and thus to zero in on a desirable solution. It thus plays a role analogous to that of temperature in simulated annealing [16].

6 Technical Leave-Behinds

AORIST has produced a number of technical insights and tools that will be valuable in future research and deployment efforts.

6.1 Technical Insights

AORIST has demonstrated several key technical insights concerning the dynamics of distributed resource allocation systems.

Resource allocation systems are rife with phase shifts.—Everywhere we turn, we find discontinuities in key system parameters. This experience reinforces the key assumption of the complexity and dynamics component of the ANT program that responsible system engineering demands attention to these behaviors. Specific examples we have encountered include the following:

- The original MG phase transition with increasing size of the strategy space persists in generalizations of the game.
- A complex phase shift occurs in a number of measures (including system utility and variance of supplier load) as the ratio of overall load to capacity varies. The shift includes coexistence regions close to but on either side of the MG configuration $N = C + 1$.
- The same load vs. capacity landscape shows several other qualitatively distinct regions. We have not had the resources to explore possible phase shifts at the boundaries of these regions.
- Every parameter of the distributed graph coloring problem we have explored can initiate phase shifts.
- The two-attractor structure of the Color RAG shows an important time-dependent dynamics with very different time constants depending on details of system initialization.

Effort profiles are algorithm-dependent.—The received wisdom in the research community dealing with constraint satisfaction and constraint relaxation is that the computational effort expended as problem size increases follows an easy-hard-easy profile for constraint satisfaction problems, but an easy-hard profile for constraint optimization problems. We have found [24] that this summary is overly simplistic. In particular, the Mini-RAG is a constraint optimization system exhibiting an easy-hard-easy profile. Exploration of other ANT systems, including CAMERA, Marbles, and the University of Kansas system, shows that in realistic systems, the effort profile depends more on the algorithm being used than on whether the system is doing constraint satisfaction or constraint optimization, and possible profiles include hard-easy as well as easy-hard and easy-hard-easy.

Pheromone learning as a control mechanism.—We have demonstrate the utility of a control mechanism inspired by insect pheromones in a wide range of applications (Sections 3.3, 4.3.5, 5.2.5). The basic pattern is that some action of an agent is determined probabilistically, by comparing a random number with a threshold. The threshold is maintained by deposit and evaporation. When the agent encounters some circumstance that would indicate it should reduce its frequency of activation, it makes a deposit on the threshold, increasing its value and thus increasing the likelihood that the next random number will be less than the threshold. Over time, the threshold evaporates by a constant fraction at each time step, thus discarding obsolete information. This mechanism is particularly useful in addressing temporal dependencies in resource allocation. In addition to deposit and evaporation, insect pheromones exhibit a third mechanism, propagation between neighboring locations. When agents are "near" one another in some topology (e.g., geographical space; resource space), the pheromone learning mechanism could be extended to include propagation. In this extension (not explored in the current project), an agent would propagate some proportion of its own no-action probability to nearby agents.

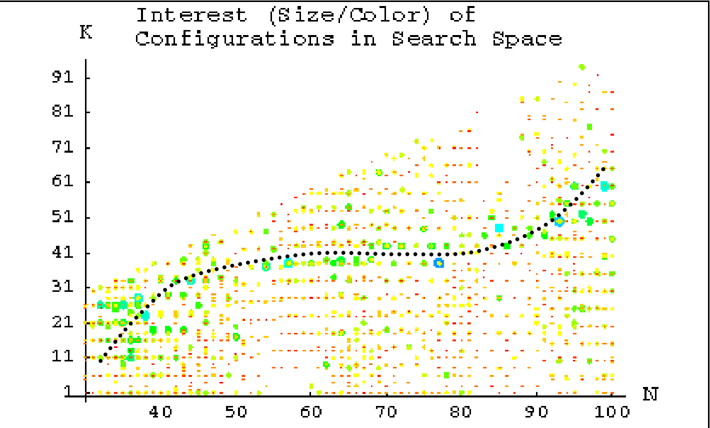


Figure 38: Result of APSE run.—The large points clustered around the dotted line represent parameter configurations of interest. A complete sweep of the parameter space would have required 121810 simulation runs. APSE found the critical region with only 20046 runs, a savings of more than 6x.

Explore vs. exploit near deadlines.—Our deadline-management mechanisms for Marbles and the Color RAG draw on a common theme that is likely to be useful in other settings as well. Resource allocation schemes in uncertain environments must include mechanisms for both *exploring* the space of possible allocations and *exploiting* assignments that appear more promising. The relative importance of exploration and exploitation is not constant

6.2 Tools

AORIST uses simulation for two purposes: *finding* regions of parameter space in which a system exhibits nontrivial behavior, and *studying* those regions in detail. The suite of software tools that we have developed for these tasks will be useful in studying the dynamics of other systems as well.

Finding interesting regions.—*Finding* interesting regions of parameter space requires multiple instantiations of a simulation [2]. The traditional approach to searching such a space is a predefined search, based on an experimental design. Factorial designs that exhaustively sweep the relevant ranges (for example, using a tool such as Drone [3]) quickly become computationally prohibitive, while designs such as Latin Squares that combine the exploration of different factors in a single run are blind to interaction effects.

To overcome these challenges, we developed the Adaptive Parameter Simulation Environment (APSE). APSE uses a derivative of Particle Swarm Optimization (PSO) [15] to search automatically through the parameter space, based on a fitness function over samples in parameter space that formalizes our specific interest (e.g., indication of phase change). Determining the fitness of any point in parameter space is very costly since it requires the repeated execution of our simulation model. APSE automatically balances the effort invested in assessing the fitness of a particular configuration with the expected reward (high fitness) by intelligently choosing the most attractive configurations for further simulation runs. Figure 38 shows how APSE is able to reduce the search space significantly as it seeks for a phase change in the distributed graph-coloring model in the ANT program. Other potential fitness functions may describe system performance boundaries whose location in a particular model would then be discovered in a PSO exploration.

Antecedents to APSE include exploratory modeling at RAND [1], the EvCA (Evolving Cellular Automata) group at the Santa Fe Institute [10] and Miller at CMU [20].

- The RAND work outlines the potential for the sort of exploration we are conducting, but does not solve the critical problem of developing fitness functions that capture dynamic phenomena of interest.
- Miller uses evolution over parameter spaces to verify models in the social sciences, seeking parameter values that break a model. In contrast, APSE searches the parameter space against two criteria: 1) What are the bounds on performance that a given approach can achieve? 2) Where might there be interesting discontinuities in behavior (e.g., phase transitions) that require further study?
- The EvCA group's use of evolution is closer to APSE. They evolve update rules for one-dimensional cellular automata to find rules that will let the automaton compute a given function of its initial state (e.g., setting all cells equal to the state of the majority of the initial cells). APSE searches for a much more complex structure.

The key to adaptive search is defining an appropriate fitness function against which to evaluate successive results. This requirement is not onerous if we are searching for performance bounds, since we simply use adaptive search to drive the performance as high (or low) as it can, and examine the slope of the performance landscape to detect leveling-off. In searching for discontinuities, the appropriate fitness function can be much more elusive. Such discontinuities are traditionally recognized by visual inspection of plots of experimental results. Our experiments in the ANT program show the promise of using the entropy of the normalized distance between sample points.

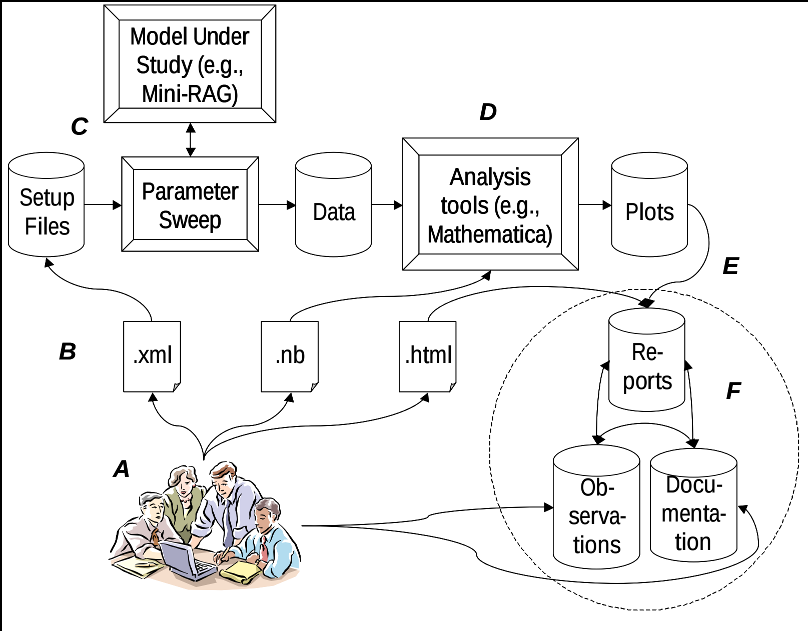


Figure 39: AISLE Framework.—Researchers (A) instantiate templates (B) to guide simulation (C), analysis (D), and result presentation (E), generating standardized HTML reports that are integrated with other research documents in an integrated online experimental notebook. Reports include links to the templates (B), ensuring repeatability. The simulations can bedistributed over different processors in a processor farm.

Studying regions in detail.—Once we identify the general region of interest we *study* its structure using AISLE (Automated Instance Sweep for Local Exploration, Figure 39), an experimental framework that enables us to configure automatic sweeps of parameter spaces and collect detailed data that feeds subsequent mathematical analysis. In contrast with APSE, AISLE executes a predetermined number of replicas (varying random seed) of each specified configuration and gathers reports generated by a wide variety of metrics. We use Mathematica as well as special-purpose Java programs to analyze and visualize the structure of the change of the model's dynamics across the chosen region of the parameter space.

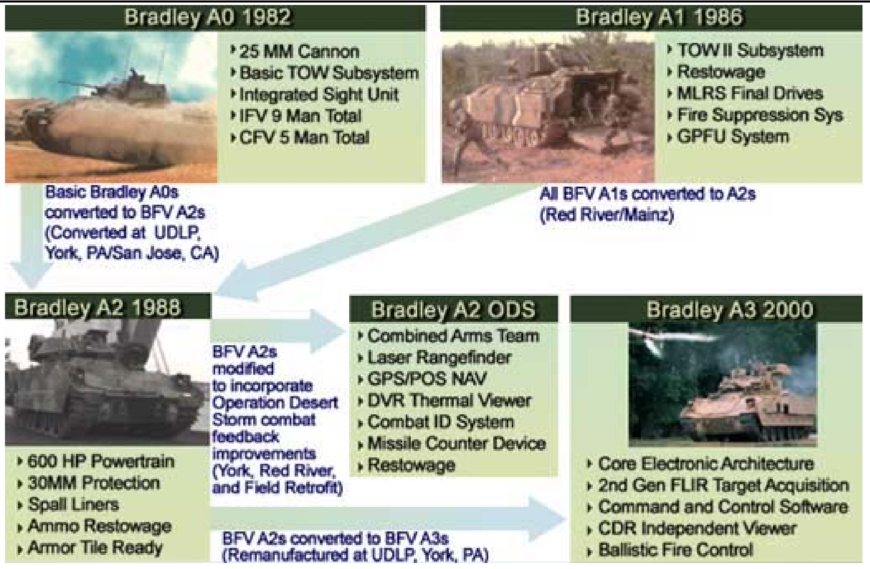


Figure 40: Migration Paths among Versions of the Bradley Fighting Vehicle

7 Transition Efforts

Altarum is vigorously pursuing transition of the AORIST technology in two main military applications: upgrading of the Army's Bradley fighting vehicle, and engineering supply networks for two DoD-sponsored programs. Both domains involve complex resource allocation problems with potentially pathological dynamics, and we are applying our AORIST tools and insights to them.

7.1 Bradley Upgrade

The Bradley fighting vehicle is the workhorse armored personnel carrier for the US Army. No new Bradleys are being manufactured, so the total population is fixed (at 6710, in varying degrees of readiness). As improved subsystems (e.g., weapons, sensors, communications, powertrain) become available, previous editions are remanufactured. Figure 40 summarizes the five main versions of the Bradley and the conversion paths among them.

All five versions of the Bradley are currently deployed to operational units. For training and maintenance reasons, all of the Bradleys in a single battalion must be of the same version. Thus battalions are upgraded all at

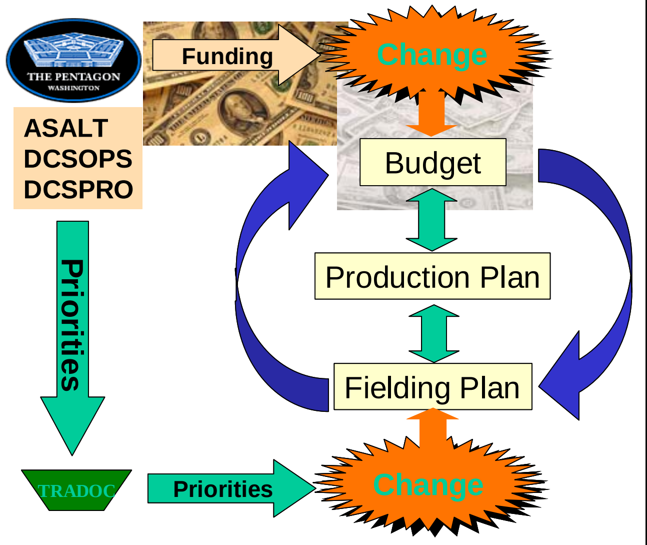


Figure 41: Conflicting Demands in Managing the Bradley

once, and a battalion is not deployable for the period of 30-60 days required to swap out an old version, swap in a new one, and train the troops on the new technology. Older Bradleys flow out of a battalion being upgraded, to battalions that are being relieved of still older models or to a remanufacturing facility, while remanufactured Bradleys flow in to take their place. This fielding plan must satisfy not only the mission objective of maximizing battalion readiness, but also capacity constraints on remanufacturing facilities, transportation resources, and the holding areas used to accumulate the set of Bradleys needed to replenish a battalion, and the total budget dollars available. These dollars are in two separate accounts that cannot be combined: acquisition (for the upgrades) and operations and maintenance (for training and deployment).

Thus development and execution of the fielding plan is a complex negotiation involving many players: multiple battalions, manufacturers, transportation vendors, and the Pentagon. To make matters worse, the Pentagon does not speak with a single voice. Figure 41 illustrates how both budget constraints and doctrinal priorities impact the system, in ways that are seldom coordinated or consistent.

The Bradley community is rife with war stories about the intractability of managing this system. The simple issue of maintaining a count of the number of each version of the Bradley currently deployed to each battalion is so challenging that the integrated spreadsheet that records this information is known as the "never-right chart."

Altarum is the sole-source supporting the Army's Bradley program office with modeling and simulation for warfighting analysis. In this capacity, we are intimately acquainted with the complexities of the fielding plan, and are often called upon to help assess the potential impact on defense readiness of adjustments to that plan. To support our customer, we are developing (at our own initiative and expense, but with the endorsement and guidance of PM Bradley) an interactive analysis tool that will model the various constraints in the upgrade process and reduce the time needed to assess the impact of a change from four hours to ten minutes. This tool will become the platform for a variety of extensions and capabilities based on AORIST.

- Our understanding of how computational effort varies as a function of load and capacity (*N x C* landscapes in the Mini-RAG) will enable us to determine when the system is near its "effective capacity," and balance leanness against the need for agility.
- Insights from our work on the impact of AL on TTS in the Color RAG suggests the dangers inherent in seeking to update information too rapidly in comparison with the inherent communications latency in the system. These dynamics are likely at least partly responsible for errors in the "never-right chart," and understanding them in more detail will yield an improved fielding plan.
- The dynamic metrics we have developed (e.g., contention, VPR, entropy measures) will enable us to project the dynamic impact of alternative fielding plans with less need for time-consuming simulations of production and shipping.
- Where simulation is necessary, APSE and AISLE will enable us to survey the space of options quickly, identifying "danger zones" to avoid.

7.2 Supply Network Engineering

Altarum (and its predecessor organizations ERIM and the Center for Electronic Commerce from ITI) has an active practice with industrial and government customers in engineering supply networks, with particular emphasis on managing their dynamical behavior. Major channels for technology transition from AORIST are through two DoD funded consortia in which we are active: ONR's Supply-chain Practices for Affordable Naval Systems (SPANS), and DLA's Defense Sustainment Consortium (DSC). Altarum is leading technical projects under both programs that will draw on tools and techniques from AORIST. The SPANS Supply Chain Dynamics project focuses on supply chains supporting NorthrupGrumman Newport News, the main shipyard supporting our nuclear carrier fleet. DSC's Robust Lean Supply Chain project will focus on manufacturing systems at Raytheon.

- The dependence of computational effort on load and capacity is again a central issue in determining effective capacity and balancing leanness against the need for agility. These models are critical in building business cases for capacity that would be reckoned "excess capacity" under traditional cost-accounting models, but that are actually necessary for dynamic stability.
- The impact of AL on TTS in the Color RAG is directly relevant to assessing optimal timing of releases in a supply chain (shipment authorizations sent from customers to their suppliers). In addition, it is likely that the dynamics of convergence depends sensitively on the topology of the supply network, and our methods may be critical in assessing the right degree of fan-out, the impact of lower-tier suppliers serving multiple mid-tier companies that converge again at the first tier, and related structural design issues.
- Our dynamic metrics will be critical for providing decision support for rough-cut capacity planning.
- APSE and AISLE can help identify "tight spots" that require more attention (e.g., improved processes, back-up stores, restructuring). In fact, AISLE is already being used in the SPANS project.

7.3 Collaboration with ISI/CAMERA

Our collaboration with the ISI team responsible for CAMERA/SNAP/Marbles has been particularly fruitful, and our mechanisms seem to offer significant improvements in the performance and efficiency of their tools. We are actively pursuing teaming opportunities where we can support them in deployment activities.

Acronyms

AL	Activation Level
ANT	Autonomous Negotiating Teams
AORIST	Agents Overcoming Resource-Independent Scaling Threats
APSE	Adaptive Parameter Search Environment
ATTEND	Automated Tools To Evaluate Negotiation Difficulty
CAg	Consumer Agent
CAMERA	Coordination and Management Environments for Responsive Agents
CBRNM	Case-Based Reflective Negotiation Model
CL	Communication latency
CS	Color Selection
DARPA	Defense Advanced Research Projects Agency

DASCh	Dynamic Analysis of Supply Chains
DLA	Defense Logistics Agency
DoC	Degree of Conflict: proportion of neighboring nodes with same color
DOC	Global Degree of Conflict
DoD	Department of Defense
DSC	Defense Sustainment Consortium
ECM	Electronic Counter-Measures
ERIM	Environmental Research Institute of Michigan
EvCA	Evolving Cellular Automata
FIP	False Information Percentage
G	Number of suppliers (colors in a colored graph)
ISI	Information Sciences Institute
ITI	Industrial Technology Institute
IXO	Information eXploitation Office
K	(Mean) number of neighbors in a graph
K-SAT	A Boolean constraint satisfaction problem with K variables per clause.
m	Length of history used for decisions by consumers
MCP	Myopic Cost-based Probability
MD	Movement Direction
MG	Minority Game
MICANT	Model Integrated Computing and Autonomous Negotiating Teams
MTS	Mean Time Series
MVP	Myopic Value-based Probability
N	Number of consumers (or nodes in a colored graph)
NBE	No-Bid Evaporation
NBP	No-Bid Probability
ONR	Office of Naval Research
OSE	Option Set Entropy
RAG	Resource Allocation Game
RIST	Resource Independent Scaling Threat
RP	Reset probability (models node failure)
S	Number of strategies per consumer
SAg	Supplier Agent
SNAP	Schedules Negotiated by Agent Planners
SPANS	Supply-chain Practices for Affordable Naval Systems
TTS	Time To Solution
UK	University of Kansas
VPR	Value Per Resource required

References

[1]S. Bankes and J. Gillogly. Exploratory Modeling: Search through Spaces of Computational Experiments. In *Proceedings of Third Annual Conference on Evolutionary Programming*, pages 353-360, World Scientific, 1994.

[2]S. C. Bankes. Tools and Techniques for Developing Policies for Complex and Uncertain Systems. In *Proceedings of RAND conference on Complex Adaptive Systems and Policy Analysis*, 2000.

[3]T. C. Belding. Drone 1.01 User's Guide. 1996. HTML, http://pscs.physics.lsa.umich.edu/Software/Drone/doc/drone.html.

[4]S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Dr.rer.nat. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.

[5]A. Bugacov. SAT encoding of the single time-slot resource allocation problem. 2001. PDF file, http://www.isi.edu/attend/Frames/Sat_encoding/sat_encoding_short_doc.pdf.

[6]A. Cavagna. Irrelevance of memory in the minority game. 1998. PS File, http://xxx.lanl.gov/ps/cond-mat/9812215.

[7]A. Cavagna, J. P. Garrahan, I. Giardina, and D. Sherrington. A thermal model for adaptive competition in a market. *Phys. Rev. Lett*, 83:4429, 1999.

[8]D. Challet and Y.-C. Zhang. Emergence of Cooperation and Organization in an Evolutionary Game. *Physica A*, 246:407, 1997.

[9]D. Challet and Y.-C. Zhang. On the Minority Game: Analytical and Numerical Studies. Institut de Physique Théorique, Université de Fribourg, Fribourg, CH, 1998. URL http://xxx.lanl.gov/ps/cond-mat/9805084.

[10] EvCA Group. Evolving Cellular Automata. 2000. Web Site, http://www.santafe.edu/projects/evca/.

[11] G. W. Flake. *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. Cambridge, MA, MIT Press, 1998.

[12] M. Frank, A. Bugacov, J. Chen, G. Dakin, P. Szekely, and B. Neches. The Marbles Manifesto: A Definition and Comparison of Cooperative Negotiation Schemes for Distributed Resource Allocation. In *Proceedings of AAAI Symposium on Negotiation Methods for Autonomous Cooperative Systems*, AAAI, 2001.

[13] T. Hogg and B. A. Huberman. Controlling Chaos in Distributed Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6 (November/December)):1325-1332, 1991.

[14] N. F. Johnson, S. Jarvis, R. Jonson, P. Cheung, Y. R. Kwong, and P. M. Hui. Volatility and Agent Adaptability in a Self-Organizing Market. *Physica A*, 256:230, 1998.

[15] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. San Francisco, Morgan Kaufmann, 2001.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671-80, 1983.

[17] Y. Li, R. Riolo, and R. Savit. Evolution in Minority Games. I. Games with a Fixed Strategy Space. Center for the Study of Complex Systems, University of Michigan, Ann Arbor, MI, 1999. URL ftp://www.pscs.umich.edu/pub/papers/evol.minority.I.v.2.0.tar.gz.

[18] Y. Li, R. Riolo, and R. Savit. Evolution in Minority Games. II. Games with Variable Strategy Spaces. Center for the Study of Complex Systems, University of Michigan, Ann Arbor, MI, 1999. URL ftp://www.pscs.umich.edu/pub/papers/evol.minority.II.tar.gz.

[19] R. Manuca, Y. Li, R. Riolo, and R. Savit. The Structure of Adaptive Competition in Minority Games. Program for the Study of Complex Systems, University of Michigan, Ann Arbor, MI, 1998. URL http://www.pscs.umich.edu/RESEARCH/pscs-tr.html.

[20] J. H. Miller. Active Nonlinear Tests (ANTs) of Complex Simulation Models. *Management Science*, 44(6 (June)):820-30, 1998.

[21] D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distribution of SAT Problems. In *Proceedings of AAAI-92*, 1992.

[22] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400(July 8):133-137, 1999.

[23] H. V. D. Parunak. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69-101, 1997.

[24] H. V. D. Parunak, S. Brueckner, J. Sauter, and R. Savit. Effort Profiles in Multi-Agent Resource Allocation. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS02)*, pages (forthcoming), 2002.

[25] H. V. D. Parunak, R. Savit, and R. L. Riolo. Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide. In *Proceedings of Multi-agent systems and Agent-based Simulation (MABS'98)*, pages 10-25, Springer, 1998.

[26] R. Savit, S. A. Brueckner, H. V. D. Parunak, and J. Sauter. General Structure of Resource Allocation Games. Altarum, Ann Arbor, MI, 2002. URL www.erim.org/~vparunak/RAGpaper.pdf.

[27] R. Savit, R. Manuca, Y. Li, and R. Riolo. The Dynamics of Minority Competition. University of Michigan Program for the Study of Complex Systems, Ann Arbor, MI, 1998. URL ftp://www.pscs.umich.edu/pub/papers/elfarol-iccs98.ps.

[28] R. Savit, R. Manuca, and R. Riolo. Adaptive Competition, Market Efficiency, and Phase Transitions. *Physical Review Letters*, 82(10):2203-2206, 1999.

Appendix A: Theoretical Analysis of MICANT-RAG (Section 4.1)

What follows is a derivation of an equation to describe a very simple version of the problem in which agents seek a resource to complete a task. We ignore any discretization of the problem and suppose that the process is continuous. This approximation, therefore, will not reproduce the oscillations seen in the simulations. We assume that there is only one resource necessary to the completion of the task, and that there is only one supplier that can supply that resource. We assume that tasks are introduced into the system continuously and at a constant rate. (Generalizations to a non-constant rate are straightforward.)

Let N be the number of open tasks. Then,

$$\frac{dN}{dt} = A - B,$$

where A is the number of new tasks introduced per unit time, and B is the number of tasks completed per unit time. To determine B, we consider the typical time between task completions. Suppose that a task is competed at time t_0. We assume that a task is completed as soon as the single resource necessary to complete the task is successfully captured by the task. After a resource is captured from the supplier, there is a refractory period, R, (a restocking time), during which no resource is available. Following the refractory period, all open tasks can bid for the resource. After a task bids for a resource, there is a grace period that must elapse before that task can bid again. If we assume that the probability for a task to bid for a resource is uniformly distributed in time, then the typical time between bids for the resource will be c/N, where c is a constant closely related to the grace period. So, if a task is completed at time t_0, (i.e. if the resource is captured by a task at time t_0, the next time a resource can be captured by a task will typically be t_0 + R + c/N. Therefore the typical time between task completions is R + c/N, and so

$$B = \frac{1}{R + \frac{c}{N}}.$$

Thus,

$$\frac{dN}{dt} = A - \frac{1}{R + \frac{c}{N}},$$

This equation is not trivial to integrate, although it may be possible to get a closed form solution. Nevertheless, we can learn some general things about it. First, let's consider the fixed point, dN/dt = 0. This occurs when

$$N = N_0 \equiv \frac{cA}{1 - AR}.$$

Note that this is unphysical unless AR < 1. In fact, this is the regime studied in our simulations. We compute R as 1/Ar, where $r \in \{1.2, 1.5, 2, 3\}$. In these terms,

$$N_0 \equiv \frac{cA}{1 - 1/r} = \frac{Arc}{r - 1}$$

If N is small (or zero) when t is small, then the number of open tasks will increase with time. N will increase until it is large enough so that resources are requested at a fast enough rate so that A=B, at which point, dN/dt=0. Thus, in this continuous approximation, the number of open tasks increases with time and asymptotes at the value N_0. This is qualitatively what we see in the simulations. Plugging in values for A, R and c, we find ratios of N_0 for different sets of values of parameters that are approximately correct. (Note that these ratios depend only on the values of AR for the different runs, not on c or on A or R, separately.)

In addition to the fast oscillations that are observed in the simulations, and that are due to the batching of tasks, this equation does not explain the apparent overshoot in N for smallish times. That is, it appears in the simulations that, depending on the values of A, the restocking time and the grace period, that N rises above its asymptotic value and approaches N_0 from above. This equation evidently does not exhibit this behavior. We believe that this overshoot is a consequence of the fact that the process (and the simulation) is discrete, and should really be modeled as a difference equation.

Appendix B: Computation of entropy for the random choice game

Here we will show that the entropy for the m-histories of states in a random choice resource allocation game, S_m, satisfies $S_m=mS_1$, where S_1 is the entropy of the distribution of outcomes for a single time step of the same game.

We consider N agents each of whom flips a G sided coin to determine which of G groups to join. We are interested in the entropy of m-strings of states.

First, at a given time step there are a total of 2^G possible states corresponding to whether each group is over or under loaded. For a given N, C and G not all of these will be accessible. Let P(j) be the probability that state j (which is one of the 2^G states) occurs in a single time step. For those states that are not accessible, P(j)=0.

P(j) can be computed in a straightforward way. Let n_k be the number of agents in group k at a given time step. Then, the probability to have the set $\{n_k\}$ is just

$$\frac{N!}{\prod_k n_k!}\left[\frac{1}{G}\right]^N \tag{1}$$

and the probability to have some state, j consisting of a G-tuple of under and over loaded groups, eg. (+,-,-,..,+), is just

$$P(j)=\sum_{\{n_k\}=L}^{\{n_k\}=U}\frac{N!}{\prod_k n_k!}\left[\frac{1}{G}\right]^N \tag{2}$$

Where $L<\{n_k\}<U$ denotes the set of values of n_k consistent with the particular state, j, whose probability is being computed. In general, there is no simple closed form for P(j). However, in the limit of large N and n_k, the P(j) are related to (and generalizations of) the error function, which are just incomplete Gaussian integrals. To see that, one needs to use Stiling's approximation: $\ln N! \approx N \ln N - N$ in the expression (2) for N! and $n_k!$. Then, the summand becomes

$$\exp\left[N\ln N - N\ln G - \sum_k n_k \ln n_k\right]. \tag{3}$$

Since this is a random choice game, we know that the mean value of n_k will be N/G. Define $N/G=\nu$, and let $n_k = \nu + r_k$. Then, (3) becomes

$$\exp\left[N\ln\nu - \sum_k \nu(1+\xi_k)\left[\ln\nu + \ln(1+\xi_k)\right]\right] \tag{4}$$

where $\xi_k \equiv r_k/\nu$.

This simplifies to

$$\exp\left[-\sum_k \nu(1+\xi_k)\ln(1+\xi_k)\right] \tag{5}$$

where we have used the fact that $\sum_k r_k = 0$ and that $\sum_k \nu \ln \nu = N \ln \nu$.

Now, we expect that r_k will be a number, typically of order $N^{1/2}$, so that ξ_k will typically be of order $N^{-1/2}$, and therefore <<1 for large N. So, we can expand (5) in a power series in ξ_k. We have

$$-\sum_k \nu(1+\xi_k)\ln(1+\xi_k) \approx -\sum_k \nu(1+\xi_k)(\xi_k - \frac{1}{2}\xi_k^2 + \ldots) \approx -\nu\sum_k \xi + \frac{1}{2}\xi_k^2 + \ldots \approx -\frac{\nu}{2}\sum_k \xi_k^2$$

where the linear term sums to zero.

So,

$$P(j) = \sum_{\{n_k\}=L}^{\{n_k\}=U} \exp\left[-\frac{\nu}{2}\sum_k \xi_k^2\right] \approx \int_{\{\xi_k=L'\}}^{\{\xi_k=U'\}} \exp\left[-\frac{\nu}{2}\sum_k \xi_k^2\right] d\xi_k \qquad (6)$$

Here the limits on the integration over the ξ_k's just reflect the same limits as in the sum. So, the P(j) are, in the limit of large N, just products and sums of error functions. In general there is no simple closed form for these, but they can be easily evaluated numerically.

From the P(j) we want to calculate the entropy for the m-strings. Since we have a RCG, the history of states, j(t) are IID. Therefore, the probability of occurrence of a particular m string is just

$$H(h_m) = \prod_{k=j_1}^{j_m} P(k) \qquad (7)$$

Where h_m is the m-string $h_m=\{j_1, \ldots, j_m\}$, and H is the probability for that m-string to occur. So, the entropy of the m-strings is just

$$S_m = \sum_{h_m} H(h_m)\ln H(h_m) = \sum_{h_m}\left\{\left[\prod_{k=j_1}^{j_m} P(k)\right]\ln\left[\prod_{k=j_1}^{j_m} P(k)\right]\right\} \qquad (8)$$

Now, consider a set of histories for which $j_1 j_2, \ldots, j_{m-1}$ are fixed. The set includes all 2^G values for j_m. Call this set β. Then,

$$\sum_{h_m\in\beta} H(h_m)\ln H(h_m) = \sum_{h_m\in\beta}\left[\prod_{k=j_1}^{j_{m-1}} P(k)\right] P(j_m)\left\{\ln\left[\prod_{k=j_1}^{j_{m-1}} P(k)\right] + \ln P(j_m)\right\} \qquad (9).$$

Summing over $h_m \in \beta$ means summing over all values of j_m. There are two terms in (9). In the first one, all factors are fixed except for $P(j_m)$. When we sum over all values of j_m, this factor sums to one, and then the first term just becomes one term that contributes to S_{m-1}. I.e., it is

$$\left[\prod_{k=j_1}^{j_{m-1}} P(k)\right]\ln\left[\prod_{k=j_1}^{j_{m-1}} P(k)\right], \qquad (10)$$

which is just the contribution of the particular m-1 string $\{j_1, \ldots, j_{m-1}\}$ to S_{m-1}. When we sum over j_m, the second term is just

$$\sum_{h_m \in \beta} \left[\prod_{k=j_1}^{j_{m-1}} P(k) \right] P(j_m) \ln P(j_m) = \prod_{k=j_1}^{j_{m-1}} P(k) \sum_{j_m} P(j_m) \ln P(j_m) = S_1 \prod_{k=j_1}^{j_{m-1}} P(k) \qquad (11)$$

where S_1 is the entropy for the m=1 string, i.e. for one time step.

Using (9), (10) and (11) in (8), we have

$$S_m = \sum_{h_{m-1}} \left\{ \left[\prod_{k=j_1}^{j_{m-1}} P(k) \right] \ln \left[\prod_{k=j_1}^{j_{m-1}} P(k) \right] + S_1 \prod_{k=j_1}^{j_{m-1}} P(k) \right\}. \qquad (12)$$

Note first that the sum here is now over values of h_{m-1} since to get (10) and (11) we summed over values of j_m. The first term in (12) is just S_{m-1} After summing over all values of h_{m-1}, the second term in (12) just becomes S_1, since

$$\sum_{h_{m-1}} \prod_{k=j_1}^{j_{m-1}} P(k) = \prod_{i=1}^{m-1} \sum_{j_i=1}^{2^G} P(j_i) = 1 \qquad (13).$$

So,

$$S_m = S_{m-1} + S_1, \qquad (14)$$

and by induction,

$$S_m = mS_1. \qquad (15)$$